

(19)



Europäisches Patentamt

European Patent Office

Office européen des brevets



(11)

EP 0 991 081 B1

(12)

EUROPEAN PATENT SPECIFICATION

(45) Date of publication and mention
of the grant of the patent:
30.11.2005 Bulletin 2005/48

(51) Int Cl.7: **G11C 16/06**

(21) Application number: **98203302.9**

(22) Date of filing: **30.09.1998**

(54) **Emulated EEPROM memory device and corresponding method**

Emulierte EEPROM Speicheranordnung und entsprechendes Verfahren

Mémoire EEPROM simulée et procédé correspondant

(84) Designated Contracting States:
DE FR GB IT

(43) Date of publication of application:
05.04.2000 Bulletin 2000/14

(73) Proprietor: **STMicroelectronics S.r.l.**
20041 Agrate Brianza (Milano) (IT)

(72) Inventors:
• **Peri, Maurizio**
24125 Bergamo (IT)
• **Brigati, Alessandro**
29015 Castel S. Giovanni (Piacenza) (IT)
• **Olivo, Marco**
24123 Bergamo (IT)

(74) Representative: **Botti, Mario**
Botti & Ferrari S.r.l.,
Via Locatelli, 5
20124 Milano (IT)

(56) References cited:
US-A- 5 651 128 **US-A- 5 796 657**

- **ATMEL: Secure Microcontrollers - Data Sheets, Secure Cryptocontrollers for Smart Cards; AT90SCC SUMMARY, updated 22 June 1998. Available from Internet: <URL:http://www.atmel.com/atmel/products/p rod153.htm> 22 February 1998. XP002094247**
- **BAHOUT Y: "Combined FLASH and EEPROM integrated circuit" ELEKTRONIK INDUSTRIE, OCT. 1997, HUTHIG, GERMANY, vol. 28, no. 10, pages 48, 50-51, XP002094246 ISSN 0374-3144**

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 0 991 081 B1

DescriptionTechnical Field

5 [0001] The present invention relates to a method and device to emulate the features of a EEPROM memory device.

[0002] More specifically, the invention relates to an integrated circuit comprising a microcontroller, a memory macrocell including a Flash EEPROM memory and means for emulating EEPROM byte alterability.

[0003] The invention relates, particularly but not exclusively, to microcontroller electronic devices having an on-board resident memory. More specifically, the device may be a microcontroller or a microprocessor having a resident (on-board) and integrated memory macrocell circuit.

10 [0004] In the embodiment being described by way of example, the memory macrocell includes an embedded Flash memory portion to store programs and update codes for the microcontroller and an embedded EEPROM non-volatile memory portion to store data.

15 Background art

[0005] As is well known, modern microcontroller are provided with on-board memory circuits to store both programs and data on the same IC.

[0006] In this specific technical field there is a felt need to have at least an EEPROM portion of the memory macrocell to be used just as a non-volatile memory for parameter storage and for defining non-volatile pointers of the stored data.

20 [0007] However, Flash and EEPROM technologies are not compatible and the higher integration degree and much lower cost of the Flash devices would suggest to realize memory macrocell including just Flash memory cells.

[0008] The memory circuit structure should comprises three portions: a main Flash memory portion, a small OTP portion and an EEPROM memory portion.

25 [0009] The Flash memory portion should include at least four sectors.

[0010] Flash and EEPROM portions have respective sense amplifiers so that one memory portion can be read while the other memory portion is written. However, simultaneous Flash and EEPROM write operations are not allowed.

[0011] Neither erasing of the EEPROM portion is possible while writing on the Flash portion.

[0012] Flash memory devices may be electrically erased by erasing the whole memory portion; while the EEPROMs may be erased on a byte by byte basis.

30 [0013] The memory macrocell has a register interface mapped in the memory area. All the operations are enabled through two control registers, one register FCR for the Flash (and OTP) portion operations and another one ECR for the EEPROM portion operations.

[0014] The status of a write operation inside the Flash portion can be monitored by a dedicated status register.

35 [0015] A known prior art solution allows the above operations by using an EEPROM software emulation addressing two Flash sectors which are dedicated to EEPROM emulation.

[0016] At each data update a pointer is added to find the new data. When a Flash sector is full all the data are swapped to the other sector. An unused sector is erased in background.

[0017] This solution presents good cycling performances in the same few bytes are continuously updated.

40 [0018] However, there are also some drawbacks which are listed hereinafter:

the best emulation is obtained by a huge managing software, at least 20Kbyte, which must be stored in the same memory circuit;

45 it might be necessary to wait for erase suspend before accessing at the EEPROM for read and write operations;

a long read access time has been experimented.

50 [0019] It is known from the Data Sheets of ATMEL - AT90SCC: Secure Crypto-controllers for Smart Cards (XP-002094247) to use an EEprom technology to implement on the same chip a byte-erasable EEprom and a sector-erasable Flash memory, having the main drawback of a long read access time.

[0020] Moreover, from the document of Bahout: "Flash/EEPROM-Kombination" (XP-002094246), Elektronik industrie, October 1997) to use a Flash technology to implement on the same chip a byte-erasable EEprom and a sector-erasable Flash memory. However, according to this document, the byte-erasability is obtained through a modification of the standard Flash array architecture.

55 [0021] A first object of the present invention is that of providing a new method for emulating an EEPROM memory portion by using a Flash memory portion.

[0022] A further object of the present invention is to provide an innovative system which allows a Flash memory

portion to emulate EEPROM byte alterability.

[0023] Another object of the present invention is that of providing microprocessor or a microcontroller having an on-board memory portion including Flash sectors emulating EEPROM byte alterability.

5 Summary of the invention

[0024] The solution idea on which the invention is based is that of providing an EEPROM hardware emulation of a Flash memory portion.

[0025] According to this solution idea, the technical problem is solved by the features of claims 1 and 8.

10 **[0026]** The feature and advantages of inventive method and device will appear from the following non-limiting description of a preferred embodiment given by way of example with reference to the annexed drawings.

Brief description of the drawings

15 **[0027]**

Figures 1 shows a schematic diagram of a memory macrocell including a Flash memory portion and an EEPROM hardware emulation according to the present invention;

20 Figure 2 shows a schematic diagram of the inside structure of the EEPROM emulated memory portion according to the invention;

Figure 3 shows a simplified and schematic view in greater detail of the EEPROM portion structure;

25 Figure 3A shows a simplified and schematic view of a register interface associated to the memory macrocell of Figure 1;

Figure 3B reports in a table form the addresses and size of each memory sector;

30 Figures 3C, 3D and 3E show a schematic view of a Flash Control Register of an EEPROM Control Register and of a Flash Status Register respectively;

Figure 4 is a high level flow-chart representing the steps of a method in accordance with the present invention;

35 Figure 4A shows a simplified and schematic view of a register interface associated to the EEPROM emulated portion of the present invention;

Figure 4B reports in a table form the addresses and size of each EEPROM memory sector;

40 Figures 4C, 4D and 4E show a schematic view of aFlash Control Register of an EEPROM Control Register and of a Flash Status Register respectively;

Figures 5 to 12 show simplified and schematic views of a series of updating phases concerning the EEPROM sectors of the memory macrocell according to the invention;

45 Figure 13 is a diagram of the write time versus the memory size for the memory of the present invention;

Figure 14 shows a simplified and schematic view of a state machine controlling an address counter inside the memory macrocell of the present invention.

50 Detailed description

[0028] With reference to the annexed drawing, with 1 is globally indicated a memory macrocell which is realized according to the present invention by a Flash EEPROM memory structure including an emulated EEPROM memory portion 2.

[0029] The memory macrocell 1 is embedded into an integrated circuit comprising also a microcontroller. The invention is specifically, but not exclusively, provided for an integrated microcontroller having an on-board non-volatile memory portion.

[0030] However, the principle of the invention may also be applied to an integrated memory circuit structure.

[0031] The memory macrocell 1 comprises a main 128 Kbyte Flash memory structure formed by a predetermined number of sectors, two of which are used to emulate EEPROM byte alterability. More specifically 8 Kbyte of the Flash memory portion are used to emulate 1 kbyte of an EEPROM memory portion.

[0032] Four sectors are provided for the Flash memory portion: a first 64 Kbyte sector F0; a second 48 Kbyte sector F1; a third 8 Kbyte sector F2 and a fourth 8 Kbyte sector F3.

[0033] A fifth 4 Kbyte sector F4 represents and corresponds to a first EEPROM emulated sector E0, while a sixth 4 Kbyte sector F5 represents and corresponds to a second emulated EEPROM sector E1.

[0034] An 8 Kbyte test portion 3 of the Flash memory macrocell 1 is provided to store test flags.

[0035] Sense amplifiers circuit portions 4 and 5 are provided at opposite sides of the memory macrocell 1, as shown in Figure 1.

[0036] Those sense amplifiers are connected to a program/erase controller 6 which cooperates with a dedicated registers interface 7 through a RAM buffer 8.

[0037] A 256 words ROM 9 is also connected to the controller 6.

[0038] The first and second EEPROM emulated sectors E0, E1 are each divided in four blocks BLOCK 0, ..., BLOCK3 of the same size. Figure 2 shows schematically the emulated EEPROM structure.

[0039] Each block is divided in up to sixtyfour pages and each page is formed by sixteen bytes.

[0040] According to the present invention, at each page update selected page data are moved to the next free block. When a sector is full, all the pages are swapped to the other sector.

[0041] Figure 3 shows a simplified and schematic view in which each block includes only four pages instead of the sixtyfour pages above mentioned. This simplified layout is used just to explain the EEPROM hardware emulation according to the invention.

[0042] Now, with specific reference also to the example of figure 5, the page updating procedure will be disclosed.

[0043] Each page inside each block must be identified to know in which block the updated page is. In this respect, a group of non-volatile pointers is used.

[0044] In each EEPROM sector E0, E1 some additional non-volatile memory locations are provided. Those locations are not accessible by the user.

[0045] Those locations are 256 Byte for each sector E0, E1 and are more than the amount strictly necessary to store the pointers. Only 66 locations are effectively used; 64 for the page pointers (one for each page) and other two for indicating the updating status of the other sector.

[0046] The above memory locations are programmed in the single bit mode (bit by bit); in other words, at each updating step different locations are programmed to "0" since in a Flash memory portion it's not possible to overwrite a memory location without a previous erasing of that location, but this would involve losing also the user's information.

[0047] The registers writing strategy must keep in consideration the fact that when a sector is erased even the registers included in that sector are erased too.

[0048] Therefore, the content of non-volatile registers is also stored in volatile memory locations to allow an efficient addressing of the EEPROM user's space.

[0049] The erasing phase is a time consuming operation for the periods of time which are normally acceptable for writing an EEPROM allocation. That's why the erasing phase is divided in a number of step corresponding to the number of blocks, which are four in this example.

[0050] In this manner the EEPROM sector complementary to the one under updating will be surely erased even in the worst case in which the same page is continuously updated. In other words, after four writing phases a swap on the other sector is required.

[0051] According to the invention, the specific erasing phase is divided in four steps providing respectively:

- a pre-programming phase to "0" of half a sector;
- a subsequent pre-programming to "0" of the other half sector;
- erasing plus erasing verify on a sample of cells;
- full erasing.

[0052] Moreover, since the EEPROM updating phase may require a certain number of steps, it has been provided for the setting of a one bit flag when the updating phase is started and for the setting of a different one bit flag when the updating phase is completed. This facilitates the recovery operation in case of a fault during updating.

[0053] Let's now consider the example of Figure 14 showing a state machine 15 (PEC) controlling an address counter 20 which receives as input control signals CTL_SIGN, INCREMENT coming from the state machine 15.

[0054] The address counter 20 is output connected to an internal address bus 21 which is inside the memory macrocell 1.

[0055] The address counter 20 doesn't correspond to the usual address counter included into a Flash memory since it receives also control signals from the state machine 15 in order to control the loading of hard-coded addresses in volatile or non-volatile registers 25. The registers 25 may be read and updated by the microcontroller during a reset phase or by the state machine 15 after an EEPROM update.

[0056] The address bus 21 is connected to the input of a 16byte RAM buffer 22 which is used for the page updating of the EEPROM. This RAM buffer 22 includes also two additional byte 23, 24 to store the page address during the page updating phase and the swap step.

[0057] When the user's program requires to write one or more byte in the EEPROM memory portion, the RAM buffer 22 is charged. Each charged memory location of the RAM buffer 22 has a supplementary bit TOBEPROG which is set so that the state machine 15 is able to complete the charging phase with "old" data in non-flagged locations just checking the content of the TOBEPROG bit during a sub-routine "Page Buffer Filling" as will be later explained.

[0058] The state machine 15 is active for instance in controlling the EEPROM page updating phase through an algorithm which will be disclosed in detail hereinafter.

[0059] Flash and EEPROM memories operations are controlled through the register interface 7 mapped in memory, see for instance the segment 22h in Figure 3A.

[0060] Flash Write Operations allows to program (from 1 to 0) one or more bytes or erase (from 0 or 1 to 1) one or more sectors.

[0061] EEPROM Write Operations allows to program (from 0 or 1 to 0 or 1) one or more bytes or erase all the memory (from 0 or 1 to 1).

[0062] Set Protection Operations allows to set Access, Write or Test Mode Protections.

[0063] As previously disclosed, the memory 1 comprises three portions: four main Flash sectors F0, F1, F2 and F3 for code, a small OTP zone included into the Flash and an EEPROM portion 2. Figure 3B reports in a table form the addresses and size of each memory sector.

[0064] The last four bytes of the OTP area (211FFCh to 211FFh) are reserved for the Non-Volatile Protection Registers and cannot be used as storage area.

[0065] The Flash memory portion, including the OTP, and the EEPROM have duplicated sense amplifiers 4, 5, so that one can be read while the other is written. However simultaneous Flash and EEPROM write operations are forbidden.

[0066] Both Flash and EEPROM memories can be fetches. Reading operands from Flash or EEPROM memories is achieved simply by using whatever microcontroller addressing mode with the Flash and in the EEPROM memory as source.

[0067] Writing in the Flash and in the EEPROM memories are controlled through the register interface 7 as explained hereinafter.

[0068] The memory macrocell 1 has a register interface 7 mapped in the memory space indicated with the segment 22h. All the operations are enabled through two control registers; A FCR (Flash Control Register) for the Flash (and OTP) operations and an ECR (EEPROM Control Register) for the EEPROM operations. Those registers are shown in Figures 3C and 3D respectively,

[0069] The status of a write operation inside the Flash memory can be monitored through a dedicated status registers: FSR (Flash Status Register) shown in Figure 3E.

1) FLASH MEMORY OPERATIONS

[0070] Four Write Operations are available for the Flash memory portion: Byte program, Page Program, Sector Erase and Chip Erase. Each operation is activated by a sequence of three instructions:

```

OR      FCR,      #OPMASK ;      Operation selection
LD      ADD,      #DATA   ;      Address and Data load
OR      FCR,      #080h  ;      Operation start

```

[0071] The first instruction is used to select the desired operation, by setting bits FBYTE, FPAGE, FSECT or FCHIP of FCR. The second instruction is used to choose the address to be modified and the data to be programmed. The third instruction is used to start the operation (set of bit FWMS of FCR).

[0072] FWMS bit and the Operation Selection bit of FCR are automatically reset at the end of the Write operation.

[0073] Once selected, but non yet started (FWMS bit still reset), one operation can be cancelled by resetting the Operation Selection bit. The eventually latched addresses and data will be reset.

[0074] In the following, when non differently specified, let's suppose than the Data Page Pointer DPR0 has been set so to point to the desired 16Kbyte block to modify, while DPR1 has been set so to point to the Register interface:

5

```

SPP      #21          ;      MMU paged registers
LD       R241,    #089h ;      Register Interface

10 LD       R240,    #000h ;      1st 16K page of Flash 0
LD       R240,    #001h ;      2nd 16K page of Flash 0
LD       R240,    #002h ;      3rd 16K page of Flash 0
LD       R240,    #003h ;      4th 16K page of Flash 0
LD       R240,    #004h ;      1st 16K page of Flash 1
15 LD       R240,    #005h ;      2nd 16K page of Flash 1
LD       R240,    #006h ;      3rd 16K page of Flash 1
LD       R240,    #007h ;      Flash 2 and Flash 3
LD       R240,    #084h ;      OTP

```

20

A) Byte Program

[0075] The Byte program operation allows to program 0s in place of 1s.

25

```

OR       0400h,    #010h ;      Set FBYTE in FCR
LD       03CA7h,   #D6h   ;      Address and Data load
30 OR       0400h,    #080h ;      Set FWMS in FCR

```

30

[0076] The first instruction is used to select the Byte Program operation, by setting bit FBYTE of FCR. The second instruction is used to specify the address and the data for the byte programming. The third instruction is used to start the operation (set of bit FWMS of FCR).

[0077] If more than one pair of address and data are given, only the last pair is taken into account. It's not necessary to use a word-wide instruction (like LDW) to enter address and data: only one byte will be programmed, but is unpredictable to know if it will be the low or the high part of the word (it depends on the addressing mode chosen).

[0078] After the second instruction the FBUSY bit of FCR is automatically set. FWMS, FBYTE and FBUSY bits of FCR are automatically reset at the end of the Byte program operation (10 μ s typical).

[0079] The Byte Program operation is allowed during a Sector Erase Suspend, and of course not in a sector under erase.

B) Page Program

[0080] The Page Program operation allows to program 0s in place of 1s. From 1 to 16 bytes can be stored in the internal Ram before to start the execution.

50

55

```

OR      0400h, #040h ; Set FPAGE in FCR
LD      028B0h, #0F0h ; 1st Address and Data
5 LD      028B1h, #0E1h ; 2nd Add and Data (Opt.)
LD      028B2h, #0D2h ; 3rd Add and Data (Opt.)

...
LD      028Bxh, #0xxh ; xth Add and Data (Opt.)

...
10 LD      028beh, #01Eh ; 15th Add and Data (Opt.)
LD      028bfh, #00Fh ; 16th Add and Data (Opt.)
OR      0400h, #080h ; Set FWMS in FCR

```

15 **[0081]** The first instruction is used to select the Page Program operation, by setting bit FPAGE of FCR. The second instruction is used to specify the first address and the first data to be programmed. This second instruction can be optionally followed by up to 15 instructions of the same kind, setting other addresses and data to be programmed. All the addresses must belong to the same page (only the four LSBs of address can change). Data contained in page addresses that are not entered are left unchanged. The third instruction is used to start the operation (set of bit FWMS of FCR).

20 **[0082]** If one address is entered more than once inside the same loading sequence, only the last entered data is taken into account. It is allowed to use word-wide instructions (like LDW) to enter address and data.

[0083] After the second instruction the FBUSY bit of FCR is automatically set. FWMS, FPAGE and FBUSY bits of FCR are automatically reset at the end of the Page Program operation (160 us typical).

25 **[0084]** The Page Program operation is not allowed during a Sector Erase Suspend.

C) Sector Erase

30 **[0085]** The Sector Erase operation allows to erase all the Flash locations to 0ffh. From 1 to 4 sectors to be simultaneously erased can be entered before to start the execution. This operation is not allowed on the OTP area. It is not necessary to pre-program the sectors to 00h, because this is done automatically.

```

35 PP      #21 ; MMU paged registers
LD      R240, #000h ; 1st 16K page of Flash 1
LD      R242, #004h ; 1st 16K page of Flash 2
LD      R243, #007h ; Flash 2 and Flash 3

```

40 **[0086]** First DPR0 is set to point somewhere inside the Flash sector 0, DPR2 inside Flash sector 1, DPR3 inside Flash sectors 2 and 3. DPR1 continues to point to the Register interface.

```

45 OR      04000h, 008h ; Set FSECT in FCR
LD      00000h, 000h ; Flash 0 selected
LD      08000h, 000h ; Flash 1 selected (Opt. 9)
LD      0C000h, 000h ; Flash 2 selected (Opt. 9)
LD      0E000h, 000h ; Flash 3 selected (Opt. 9)
50 OR      04000h, 080h ; Set FWMS in FCR

```

55 **[0087]** The first instruction is used to select the Sector Erase operation, by setting bit FSECT of FCR. The second instruction is used to specify an address belonging to the first sector to be erased. The specified data is don't care. This second instruction can be optionally followed by up to three instructions of the same kind, selecting other sectors to be simultaneously erased. The third instruction is used to start the operation (set of bit FWMS of FCR).

[0088] Once selected, one sector cannot be deselected. The only way to deselect the sector, it to cancel the operation, by resetting bit FSECT. It is allowed to use word-wide instructions (like LDW) to select the sectors.

[0089] After the second instruction the FBUSY bit of FCR is automatically set. FWMS, FSECT and FBUSY bits of FCR are automatically reset at the end of the Sector Erase operation (1,5 s typical).

[0090] The Sector Erase operation can be suspended in order to read or to program data in a sector not under erase. The Sector Erase operation is not allowed during a Sector Erase Suspend.

C.1) Sector Erase Suspend/Resume

[0091] The Sector Erase Suspend is achieved through a single instruction.

OR 0400h, #004h ; Set FSUSP in FCR

[0092] This instruction is used to suspend the Sector Erase operation, by setting bit FSUSP of FCR. The Erase Suspend operation resets the Flash memory to normal read mode (automatically resetting bits FWMS and FBUSY) in a maximum time of 15us. Bit FSECT of FCR must be kept set during the Sector Erase Suspend phase: if it is software reset, the Sector Erase operation is cancelled and the content of the sectors under erase is not guaranteed.

[0093] When in Sector Erase Suspend the memory accepts only the following operations: Read, Sector Erase Resume and Byte program. Updating the EEPROM memory is not possible during a Flash Sector Erase Suspend.

[0094] The Sector Erase operation can be resumed through two instructions:

AND 04000h, #0FBh ; Reset FSUSP in FCR
OR 04000h, #080h ; Set FWMS in FCR

[0095] The first instruction is used to end the Sector Erase Suspend phase, by resetting bit FSUSP of FCR. The second instruction is used to restart the suspended operation (set of bit FWMS of FCR). After this second instruction the FBUSY bit of FCR automatically set again.

D) Chip Erase

[0096] The Chip Erase operation allows to erase all the Flash locations to 0ffh. This operation is simultaneously applied to all the 4 Flash sectors (OTP area excluded). It is not necessary to pre-program the sectors to 00h, because this is done automatically.

OR 04000h, #020h ; Set FCHIP in FCR
OR 04000h #080h ; Set FWMS in FCR

[0097] The first instruction is used to select the Chip Erase operation, by setting bit FCHIP of FCR. The second instruction is used to start the operation (set of bit FWMS of FCR).

[0098] It is not allowed to set the two bits (FCHIP and FWMS) with the same instruction.

[0099] After the second instruction the FBUSY bit of FCR is automatically set. FWMS, FCHIP and FBUSY bits of FCR are automatically reset at the end of the Chip Erase operation (3 s typical).

[0100] The Chip Erase operation cannot be suspended. The Chip Erase operation is not allowed during a Sector Erase Suspend.

2) EEPROM MEMORY OPERATIONS

[0101] Two Write Operations are available for the EEPROM memory: Page Update and Chip Erase. Each operation is activated by a sequence of three instructions:

```

OR      ECR,      #OPMASK ;      Operation selection
LD      ADD,      #DATA   ;      Address and Data load
OR      ECR,      #080h   ;      Operation start

```

[0102] The first instruction is used to select the desired operation, by setting bits EPAGE or ECHIP of ECR. The second instruction is used to choose the address to be modified and the data to be programmed. The third instruction is used to start the operation (set of bit EWMS of ECR).

[0103] EWMS bit and the Operation Selection bit of ECR are automatically reset at the end of the Write operation.

[0104] Once selected, but not yet started (EWMS bit still reset), one operation can be cancelled by resetting the Operation Selection bit. The eventually latched addresses and data will be reset.

[0105] In the following, when not differently specified, let's suppose that the Data Page Pointer DPR0 has been set so to point to the EEPROM to modify, while DPR1 has been set so to point to the Register interface:

```

SPP     #21                ;      MMU paged registers
LD      R241, #089h        ;      Register Interface

LD      R240, #088h        ;      EEPROM

```

[0106] It's important to note that the EEPROM operations duration are related to the EEPROM size, as shown in the table of Figure 4B.

A) Page Update

[0107] The page Update operation allows to write a new content. Both 0s in place of 1s and 1s in place of 0s. From 1 to 16 bytes can be stored in the internal Ram buffer before to start the execution.

```

OR      04001h, #040h      ;      Set EPAGE in ECR
LD      001C0g, #0F0h      ;      1st Address and Data
LD      001C1h, #0E1h      ;      2nd Add and Data (opt.)
LD      001C2h, #0D2h      ;      3rd Add and Data (opt.)
...
LD      001Cxh, #0xxh      ;      xth Add and Data (opt.)
...
LD      001ceh, #01Eh      ;      15th Add and Data (opt.)
LD      001cfh, #00Fh      ;      16th Add and Data (opt.)
OR      04001h, #080h      ;      Set EWMS in ECR

```

[0108] The first instruction is used to select the Page Update Operation, by setting bit EPAGE of EVR. The second instruction is used to specify the first address and the first data to be modified. This second instruction can be optionally followed by up to 15 instructions of the same kind, setting other addresses and data to be modified. All the addresses must belong to the same page (only the four LSBs of address can change). Data contained in page addresses that are not entered are left unchanged. The third instruction is used to start the operation (set of bit EWMS of ECR).

[0109] If one address is entered more than once inside the same loading sequence, only the last entered data is taken into account. It is allowed to use word-wide instructions (like LDW) to enter address and data.

[0110] After the second instruction the EBUSY bit of ECR is automatically set. EWMS, EPAGE and EBUSY bits of ECR are automatically reset at the end of the Page Update operation (30 ms typical).

[0111] The Page Update operation is not allowed during a Flash Sector Erase Suspend.

B) Chip Erase

[0112] The Chip Erase operation allows to erase all the EEPROM locations to 0ffh.

5

```

OR      04001h,  #020h    ;      Set ECHIP in ECR
OR      04001h,  #080h    ;      Set EWMS in ECR

```

10 **[0113]** The first instruction is used to select the Chip Erase operation, by setting bit ECHIP of ECR. The second instruction is used to start the operation (set of bit EWMS of ECR).

[0114] It is not allowed to set the two bits (ECHIP and EWMS) with the same instruction.

[0115] After the second instruction the EBUSY bit of ECR is automatically set. EWMS, ECHIP and EBUSLY bits of ECR are automatically reset at the end of the Chip Erase operation(70 ms typical).

15 **[0116]** The Chip Erase operation cannot be suspended. The Chip Erase operation is not allowed during a Flash Sector Erase Suspend.

3) Protections Operations

20 **[0117]** Only one Write Operation is available for the Non Volatile Protection Registers: Set Protection operation allows to program 0s in place of 1s. From 1 to 4 bytes can be stored in the internal Ram buffer before to start the execution. This operation is activated by a sequence of 3 instructions:

```

25 OR      FCR,      #002h    ;      Operation selection
LD      ADD,      #DATA    ; Address and Data load
OR      FCR,      #080h    ; Operation start

```

30 **[0118]** The first instruction is used to select the Set protection operation, by settling bit PROT of FCR. The second instruction is used to specify the first address and the first data to be programmed. This second instruction can be optionally followed by up to three interactions of the same kind, setting other addresses and data to be programmed. All the addresses must belong to the Non Volatile Protection registers (only the two LSBs of address can change). Protection Registers contained in addresses that are not entered are left unchanged. Content of not selected bits inside selected addresses are left unchanged, too. The third instruction is used to start the operation (set of bit FWMS of FCR).

35 **[0119]** If one address is entered more than once inside the same loading sequence, only the last entered data is taken into account. It is allowed to use word-wide instructions (like LDW) to enter address and data.

[0120] After the second instruction the FBUSY bit of FCR is automatically set. FWMS, PROT and FBUSY bits of FCR are automatically reset at the end of the Set protection operation (40 µs typical).

40 **[0121]** Once selected, but not yet started (FWMS bit still reset), the operation can be cancelled by resetting PROT bit. The eventually latched addresses and data win be reset.

[0122] The Set Protection operation is not allowed during a Sector Erase Suspend.

[0123] In the following, when not differently specified, let's suppose that the Data Page pointer DPR0 has been set so to point to the OTP area to modify, while DPR1 has been set so to point to the Register interface:

45

```

      SPP      #21      ;      MMU paged registers
LD      R241,    #089h ;      Register Interface
50      LD      R240,    #084h ;      OTP

```

[0124] There are three kinds of protections: access protection, write protections and test modes disabling.

55 **3.1) Non Volatile Registers**

[0125] The protection bits are stored in the last four locations of the OTP area (from 211FFCh to 211FFFh), as shown in Figure 4A. All the available protections are forced active during reset, then in the initialisation phase they are read

from the OTP area.

[0126] The four Non Volatile Registers used to store the protection bits for the different protection features are one Time Programmable.

[0127] The access to these registers is controlled by the protections related to the OTP area where they are mapped.

3.2) Set Access Protections

[0128] The Access Protections are given by bits APRA, APRO, APBR, APEE, APEX of NVAPR.

```

OR      04000h, #002h ; Set PROT in FCR
LD      01FFCh, #0F1h ; Prog WPRS3-1 in NVWPR
OR      04000h, #080h ; Set FWMS in FCR

```

[0129] The first instruction is used to select the Set Protection operation, by setting bit PROT of FCR. The second instruction is used to specify the NVAPR address and the new protection content to be programmed. The third instruction is used to start the operation (set of bit FWMS of FCR).

3.3) Set Write Protections

[0130] The Write Protections are given by bits WPBR, WPEE, WPRS3-0 of NVWPR.

```

OR      04000h, #002h ; Set Prot in FCR
LD      01FFDh, #0F1h ; Prog WPRS3-1 in NVWPR

```

```

OR      04000h, #080h ; Set FWMS in FCR

```

[0131] The first instruction is used to select the Set Protection operation, by setting bit PROT of FCR. The second instruction is used to specify the NVWPR address and the new protection content to be programmed. The third instruction is used to start the operation (set of bit FWMS of FCR).

[0132] The Write Protections can be temporary disabled by executing the Set Protection operation and writing 1 into these bits.

```

OR      01000h, #002h ; Set Prot in FCR
LD      01FFDh, #0F2h ; Prog WPRS0 in NVWPR
OR      01000h, #080h ; Temp Unprotected WPRS1
OR      01000h, #080h ; Set FWMS in FCR

```

[0133] The Non Volatile content of the temporary unprotected bit remains unchanged, but now the content of the temporary unprotected sector can be modified.

[0134] To restore the protection it needs to reset the micro or to execute another Set protection operation and write 0 into this bit.

3.4) Disable Test Modes

[0135] The Test Mode Protections are given by bits TMDIS and PWOK of NVWPR.

```

OR      04000h,      #002h      ; Set PROT in FCR
LDW     01FFEh,      #05A7Ch    ; Prog NVPWD1-0
OR      04000h,      #080h      ; Set FWMS in FCR

```

5

[0136] The first instruction is used to select the Set Protection operation, by setting bit PROT of FCR. The second instruction must be word-wide and it is used to specify the NVPWD1-0 address and the password to be programmed. The third instruction is used to start the operation (set of bit FWMS of FCR). The second instruction automatically forces TMDIS bit of NVWPR to be programmed.

10

[0137] Once disabled the Test Modes can be enabled again only by repeating the disable test mode sequence with the right Password. If the given data is matching with the programmed password in NVPWD1-0, bit PWOK of NVWPR is programmed and Test Modes Are enabled again.

15

[0138] If the given data is not matching, one of bits PWT2-0 of NVAPR is programmed. After three unmatching trials, when all bits PWT2-0 are programme, Test Modes are disabled for ever.

[0139] Just as an example, hereinafter a program erase controller algorithm for the Flash/EEPROM macrocell 1 is reported. This algorithm uses a call subroutine instructions named CAL and return from subroutine instructions named RET with four nested levels allowed.

20 Available Instructions:

[0140]

```

25  ALT      input      ; Wait for input at 1
    CMP      input      ; Compare input and set a Flag if 1 (x3 instructions:
                           three CMPi are existing, CMP1, CMP2,CMP3)
    JMP      label      ; Jump to label
    JIF      label      ; Jump to label if Flag = 1
    JFN      label      ; Jump to label if Flag = 0
30  CAL      label      ; Call subr. (Store PC, Inc. SP and Jump to label)
    CLF      label      ; Call subr. if Flag = 1
    RET      ; Return from subr. (Dec. SP and Jump to stored PC)
    STO      output     ; Set output at 1 (x5 instructions: 5 STOi instr. are
                           existing, STO1, STO2, ... STO5) {this instr. is used to activate any
35  Output signal of the PEC};

```

40

45

50

55

Input Variables:

[0141]

```

5      NOTHING      = No variables => Realize a NOP with CMP NOTHING;
      ALL0          = All0 phase active
      ALLERASED     = All sectors erased
      DATOOK        = Data verified equal to the target
10     ENDPULSE     = End of Prog or Erase pulse
      ERSUSP        = Erase Suspended
      LASTADD       = Last Row or Column
      LASTSECT      = Last Sector
      MAXTENT       = Reached maximum tentative number allowed
      NORMOP        = Normal Read conditions restored
15     SOFTP        = Soft Program phase active
      SUSPREQ       = Erase Suspend request pending
      TOBEMOD       = Sector to be erased or byte to be programmed
      VPCOK         = Verify voltage reached by Vpcx;

20     BYTERCY_FF   = Flash Byte Prog operation active or RECYCLE test mode
      CHIPER_EE     = EEPROM Chip Erase operation active
      CSERASE_FF    = Flash Sector/Chip Erase operation active
      NEWERPH0      = Erase phase 1-3 active
      NEWERPH1      = Erase phase 2-3 active
      NEEDERASE     = Unused sector erase needed
25     NEEDSWAP     = Sector Swap needed
      PAGEPG_EE     = EEPROM Page Update operation active
      PAGENSP_FF    = Flash Page Prog operation active or NOSOFTPtest mode
      SELPAGE       = Selected Page to update
      SWAPFAIL      = Swap error => autosuspend needed;

```

Output Variables:

[0142]

```

35     NOTHING      = No variables => Used to reset other variables;
      ALL0          = Start/Stop All0 phase (toggle)
      CUIRES        = Reset Command Interface and PEC
      ERASE         = Start/Stop Erase phase (toggle)
40     HVNEG        = Start Erase pulse
      INCCOLM       = Increment Column Address
      INCROW        = Increment Row Address
      INCSECT       = Increment Sector Address
      INCTENT       = Increment tentative number
      LOADADD       = Load column address from RAM BUFFER
45     LOADSECT     = Load sector address from RAM BUFFER
      PROGRAM       = Start Prog pulse
      READSUSP      = Stop the clock during erase suspend
      RESFLAG       = Eliminate current sector from the list to be erased
      SOFTP         = Start/Stop Soft Program phase (toggle)
50     STOREADD     = Store column address in RAM BUFFER
      SWXATVCC      = Switch Vpcx at Vcc (read voltage)
      VERIFY        = Set Verify mode
      VPCYON        = Switch On/Off Vpcy pump (toggle);

55     END_SWAP     = Reset NEEDSWAP (toggle)
      FORCESWAP     = Force NEEDSWAP=1 (toggle)
      INCPAGE       = Increment Page address
      LDDATA        = Load data from RAM buffer

```

```

LDNVCSS      = Load NVCSS address (from hardware)
LDNVESP      = Load NVESP address (from hardware)
LDOLDSECT    = Load Old sector address (from hardware)
5  LDPAGE     = Load Page address from RAM BUFFER
LDPAGE2      = Load Page address from RAM BUFFER (for Sector Swap)
LDVCSS       = Load VCSS address (from hardware)
LDVESP       = Load VESP address (from hardware)
PAGE         = Start/Stop Page Program phase (toggle)
10  READ      = Set/Reset read conditions (toggle)
STOREDATA    = Store read data into the RAM buffer
STOREPAGE    = Store page address in RAM BUFFER
STOREPAGE2   = Store page address in RAM BUFFER (for Sector Swap)
STOREPROT    = Store Protection data into the RAM buffer
STORESECT    = Store sector address in RAM BUFFER
15  SWAP      = Set/Reset Sector Swap phase (toggle)
WRITEVS      = Write Volatile Status;

```

Possible Operations:

20 [0143]

```

Flash Byte Program      (1 nesting level)
Flash Page Program      (2 nesting levels)
25 Flash Chip/Sector Erase (3 nesting levels)
Flash Byte Program while Erase Suspend (4 nesting levels)
Set Protections         (2 nesting levels)

EEPROM Page Update      (4 nesting levels)
30 EEPROM Chip Erase    (4 nesting levels)

```

CODE SIZE = 251 lines;

35

[0144] In this example, only EEprom Page Update will be described

MAIN PROGRAM:

40 [0145]

```

45  CMP      PAGEPG_EE      ;      EEPROM Chip Update op. selected ?
    JIF      epgupd        ;      If yes jump to EEPROM Chip update routine
    JMP      main          ;      If no, then loop

```

50

55

SUBROUTINES:

1) SDELAY (the PEC clock is used to count a delay for analog signals settling)

5 [0146]

```

          CMP     NOTHING      ; NOP: delay cycle
          CMP     NOTHING      ; NOP: delay cycle
10         CMP     NOTHING      ; NOP: delay cycle
          CMP     NOTHING      ; NOP: delay cycle
          RET                               ; 4 NOP + 1 CAL + 1 RET = 6 NOP

```

15 2) PROGRAM 1 BYTE (every byte is continuously programmed up to a positive verify test)

[0147]

```

20         sbytepg
          STO     VERIFY        ; Verify Data to be programmed
          CMP     DATOOK        ; Compare read data with 00h
          JIF     sbpend        ; If DATOOK=1 => Return (the data is already OK)
          STO     PROGRAM      ; If DATOOK=0 => Apply Prog pulse
          ALT     ENDPULSE      ; Wait for end of Prog pulse
25         STO     INCTENT      ; If no Increment tentative number
          CMP     MAXTENT      ; Compare tentative number with maximum allowed
          JFN     sbytepg      ; If MAXTENT=0 => Retry
          sbpend RET          ; If MAXTENT=1 || DATOOK=1 => Return

```

30 3) PROGRAM 1 PAGE

[0148]

```

35         spagepg
          STO     LDDATA        ; Read Data and flag TOBEPROG from RAM buffer
          CMP     TOBEMOD      ; Byte to be programmed ?
          JFN     sppincc      ; If no increment column
          sppbyte
          CAL     sbytepg      ; Byte Program
40         sppincc
          STO     INCCOLM      ; Increment Column address
          CMP     LASTADD      ; Last column ?
          JFN     spagepg      ; If no restart program
          RET                 ; If yes Return

```

45

50

55

4) PROGRAM 1 SECTOR

[0149]

```

5          sssectpg
          CAL      sbytepg      ; Byte Program
          STO      INCROW       ; Increment row
          CMP      LASTADD      ; Last Row ?
          JFN      sssectpg     ; If no continue All0
10         CMP      NOTHING     ; NOP: delay cycle
          STO      INCCOLM      ; Increment Column address
          CMP      LASTADD      ; Last column ?
          JFN      sssectpg     ; If no program again
          RET              ; If yes Return

```

5) ERASE VERIFY 1 SECTOR (the sector is erased, read and verified byte after byte and after every erasing pulse starting from the last non erased byte; the subroutine terminates when the last byte of the sector is erased)

[0150]

```

20          servfy
          STO      VERIFY       ; Verify Data to be erased
          CMP      DATOOK       ; Compare read Data with 0FFh

25

          JFN      sevend       ; If DATOOK=0 => Return
          STO      INCROW       ; If DATOOK=1 => Increment Row
          CMP      LASTADD      ; Last Row ?
30         JFN      servfy      ; If no continue Erase Verify phase
          STO      INCCOLM      ; If yes increment Column address
          CMP      LASTADD      ; Last Column ?
          JFN      servfy      ; If no continue Erase Verify phase
          STO      RESFLAG      ; If yes the current sector is erased
35         sevend  RET          ; Return

```

EEPROM ROUTINES

[0151] 1) PAGE BUFFER FILLING. This routine is used to fill all not selected addresses of the selected page with the old data written in those locations. Old Data are read from the old locations (using actual EESECT and EEBCK<1: 0>, the Volatile registers) using normal read conditions (Vpcx=4.5V) forced through signal READ.

[0152] Once Stored the old data in Ram, the local flag TOBEPROG for that byte is automatically set.

```

45         ebuffil
          STO      READ         ; Enter Read mode conditions
          ebffloop
          STO      LDDATA       ; Read flag TOBEPROG from RAM buffer
          CMP      TOBEMOD      ; Byte to be programmed ?
          JIF      ebfincc      ; If yes increment column
50         STO      VERIFY      ; If no Read Old Data (ST03)
          STO      STOREDATA    ; Store Old Data in Ram Buffer (ST02)
          ebfincc
          STO      INCCOLM      ; Increment Column address (ST05)
          CMP      LASTADD      ; Last column ?
          JFN      ebffloop     ; If no continue RAM filling
55         STO      READ         ; If yes exit Read mode
          RET              ; Return

```

[0153] 2) NON VOLATILE STATUS PROGRAM. This routine is used to program the Non Volatile Status Pointers

NVESP, NVCSS0, NVCSS1.

```

5          estprg
          CAL      sbytepg      ; Non Volatile Status Program
          CMP      NOTHING      ; NOP: delay cycle
          RET                          ; Return

```

[0154] 3) PAGE PROGRAM. This routine is used to program a Page taking the data from the RAM buffer. At first not selected page address in the Ram buffer are filled with the last valid data. Then the VIRG bit in NVESP is programmed to notify that the page program operation is started. Then after the page programming, the USED bit in NVESP is programmed to notify that the operation is concluded. At the end also the Volatile Block Pointers are updated

```

15          epagepg
          STO      STOREPAGE      ; Store current Page Address in RAM
          CAL      ebuffil      ; Fill-in not selected page address
          STO      LDNVESP      ; Load New NVESP address for current page
          CAL      estprg      ; NVESP Program (VIRG bit)
          STO      LDPAGE      ; Load New Page address from RAM
20          CAL      spagepg      ; Page Program
          STO      LDNVESP      ; Load New NVESP address for current page
          CAL      estprg      ; NVESP Program (USED bit)
          STO      LDVESP      ; Load VESP address for current page (STO2)
          STO      WRITEVS      ; Write Volatile Status BCK<1:0> (STO3)
25          RET                          ; Return

```

[0155] 4) SECTOR SWAP. This routine is used when in the current sector the 4 blocks for the selected page are already used. In this case the selected page is programmed in the new sector and all the other unselected pages must be swapped to the new sector.

[0156] SWAP=1 forces TOBEPROG=0 => in ebuffil routine all the Page data are copied into the RAM buffer

[0157] CHIPER EE=1 forces TOBEPROG=1 => in ebuffil routine all the data in the Ram buffer are kept at FFh (reset value).

[0158] CHIPER_EE=1 forces SELPAGE=0 => no page selected

```

35          esecswp
          STO      SWAP,PAGE ; Enter Sector Swap (STO4)
          esspage
          STO      LDPAGE2      ; Read selected page address from RAM (STO1)
          CMP      SELPAGE      ; Current page is the selected for update ?
40          JIF      essincp      ; If yes increment page
          CAL      epagepg      ; If no Page Program

```

```

45          essincp
          STO      INCPAGE      ; Increment Page
          CMP      LASTADD      ; Last Page ?
          JFN      esspage      ; If no swap current page
          CMP      SWAPFAIL      ; Swap fail ?
50          JIF      sexit      ; If yes autosuspend
          STO      SWAP,PAGE      ; Exit Sector Swap phase
          RET                          ; Return

```

[0159] 5) PROGRAM ALLO. This routine is used for program A110 This routine automatically program bit ACT of unused sector when the sector swap is done.

```

eall0
STO     ALLO           ; Enter All0 phase (STO4)
STO     LDOLDSECT      ; Load Old Sector address (STO1)
5      CAL     ssectpg   ; Sector Program
STO     ALLO           ; Exit All0 phase
RET     ; Return

```

[0160] 6) ERASE. This routine is used for erase. Erase verify is made before the first erase pulse, since during Erase phase 3, the initial cells status is unknown.

```

eerase
STO     ERASE          ; Enter Erase phase (STO4)
15     STO     LDOLDSECT ; Load Old Sector address (STO1)
eervfy
CAL     servfy         ; Erase Verify on all sector
CMP     ALLERASED      ; All sector erased ?
JIF     eerend         ; If yes exit erase phase
eerpul
20     STO     HVNEG     ; If no apply Erase pulse
ALT     ENDPULSE       ; Wait for end of Erase pulse
CMP     NOTHING        ; NOP: reset the counter when HVNEG=1
STO     INCTENT        ; Increment tentative number
CMP     MAXTENT        ; Compare tentative number with maximum
25     allowed
JFN     eervfy         ; If MAXTENT=0 => erase verify
eerend
STO     ERASE          ; If MAXTENT=1 => exit Erase phase
RET     ; Return

```

[0161] 7) SECTOR ERASE. This routine is used to enter the needed erase phase on the unused sector, as explained by the following table:

| EPH<3:0> | EEERPH<1:0> | NEWERPH<1:0> | NEEDSWAP | NEEDERASE | Er.Phase |
|------------|-------------|--------------|----------|-----------|----------|
| 0000 | 11 | 00 | 0 | 0 | None |
| 0000(1111) | 11 | 00 | 1 | 1 | 0 |
| 1110 | 00 | 01 | 0 | 1 | 1 |
| 1100 | 01 | 10 | 0 | 1 | 2 |
| 1000 | 10 | 11 | 0 | 1 | 3 |

[0162] Erase phase 0 makes the A110 on the second half (status included) (second half is programmed first just because it includes at the end of its 'address space' the NV status, whose bits must be programmed as soon as possible)

Erase phase 1 makes the A110 on the first half.

Erase phase 2 makes the Erase with verification of status only

Erase phase 3 completes the Erase

```

esector
STO    LDNVCSS      ; Load NVCSS0 address
CAL    estprg       ; NVCSS0 Program (bit EPHS<3:0>)
CMP    NEWERPH1     ;
5     JFN    eseph01 ; If NEWERPH<1:0>=0X => Enter Erase Phase 0-1
CAL    eerase       ; If NEWERPH<1:0>=1X => Enter Erase Phase 2-3
CMP    NEWERPH0     ;
JFN    eseend       ; If NEWERPH<1:0>=10 => Exit Erase Phase 2-3
eseph01
10    CAL    eall0   ; Program All0 (needed before any erase)
eseend
STO    LDNVCSS      ; Load NVCSS0 address
CAL    estprg       ; NVCSS0 Program (bit EPHE<3:0>)
STO    LDVCSS       ; Load VCSS0 address (STO2)
STO    WRITEVS      ; Write Volatile Status ERPH<1:0> (STO3)
15    CMP    NEEDSWAP ;
JFN    eseret       ; If NEEDSWAP=0 => exit Sector Erase
STO    ENDSWAP      ; If NEEDSWAP=1 => ENDSWAP resets NEEDSWAP
JMP    eseend       ; Program NVCSS1 (CUR bit) and VCSS1 (EESect)
eseret
20    RET           ; Return

```

[0163] 8) PAGE UPDATE. This routine is used to handle all the data transfers between blocks and sectors when an update of a page of the EEPROM is needed

```

25    epgupd
ALT    VPCOK        ; Wait for Vpcx verify voltage (was Read mode)
STO    STOREPAGE2   ; Store Page address in RAM (Sect Swap) (STO2)
STO    PAGE         ; Enter Page Program phase (STO4)
CAL    epagepg      ; Selected Page Program
30    STO    PAGE     ; Exit Page Program phase
CMP    NEEDSWAP     ; EEPROM Sector Swap needed ?
CLF    esecswp      ; If yes Sector Swap
CMP    NEEDERASE     ; Unused Sector Erase needed ?
CLF    esector      ; Unused Sector Erase
35
JMP    sexit        ; Exit Page Update

```

40 [0164] The memory device and the method according to the invention allow a totally hardware emulation of an EEPROM memory portion.

[0165] No access differences are detectable between the emulated memory portion according to the invention and a standard EEPROM memory.

45 [0166] An immediate EEPROM access is available during the reading and writing phases of the emulated memory portion 2.

[0167] A further advantage is given by the Flash code execution running during EEPROM modify phase.

Claims

50 1. Integrated circuit comprising a microcontroller, a memory macrocell (1) including a Flash EEPROM memory formed by a predetermined number of sectors (F0, F1, F2, F3, F4, F5) and means for emulating the features of a EEPROM memory device incorporated into said memory macrocell (1); , wherein said means comprises at least two sectors (E0, E1) of said Flash EEPROM memory that are adapted to be used to emulate EEPROM byte alterability by

55 dividing each of said two sector in a pre-determined number of blocks (BLOCK 0, ..., BLOCK3) of the same size and each block in a predetermined number of pages and updating the emulated EEPROM memory portion by programming different memory locations in a single bit mode so that at each page update selected page data are moved to the next free block and, when data are written into each one of the blocks of said first sector, all the

pages are swapped to said second sector.

2. Integrated circuit according to claim 1, **characterized in that** said EEPROM byte alterability is emulated by hardware means.
3. Integrated circuit according to claim 1, **characterized in that** 8 Kbyte of the Flash memory portion are used to emulate 1 kbyte of an EEPROM memory portion.
4. Integrated circuit according to claim 1, **characterized in that** a state machine (15) is provided for controlling an address counter (20) which is output connected to an internal address bus (21) running inside said memory macrocell (1), said address counter (20) receiving control signals from the state machine (15) in order to control the loading of hard-coded addresses in volatile or non-volatile registers (25) which are read and updated by the microcontroller during a reset phase or by the state machine (15) after an EEPROM update.
5. Integrated circuit according to claim 4, **characterized in that** said address bus (21) is connected to the input of a RAM buffer (22) which is used for the page updating of the EEPROM including two additional byte (23, 24) for storing the page address during a page updating phase.
6. Integrated circuit according to claim 1, **characterized in that** Flash and EEPROM memories operations are controlled through a register interface (7) mapped into the memory (1).
7. Method for emulating the features of a EEPROM memory device incorporated into a memory macrocell (1) which is embedded into an integrated circuit comprising also a microcontroller and including a Flash EEPROM memory formed by a predetermined number of sectors (F0, F1, F2, F3, F4, F5), at least two sectors (E0, E1) of the Flash memory structure being used to emulate EEPROM byte alterability by dividing each of said two sectors in a predetermined number of blocks (BLOCK 0, ..., BLOCK3) of the same size and each block in a predetermined number of pages and updating the emulated EEPROM memory portion by programming different memory locations in a single bit mode, wherein at each page update selected page data are moved to the next free block and, when data have been written into each one of the blocks of said first sector, all the pages are swapped to said second sector.

Patentansprüche

1. Integrierte Schaltung mit einem Mikrocontroller, einer Speichermakrozelle (1) mit einem Flash-EEPROM-Speicher, der durch eine vorbestimmte Anzahl von Sektoren (F0, F1, F2, F3, F4, F5) gebildet ist, und einer Einrichtung zum Emulieren der Leistungsmerkmale eines EEPROM-Speicherbausteins, der in der Speichermakrozelle (1) enthalten ist; wobei die Einrichtung mindestens zwei Sektoren (E0, E1) des Flash-EEPROM-Speichers umfasst, die so ausgelegt sind, dass sie zum Emulieren der EEPROM-Byte-Veränderlichkeit verwendet werden können, indem die beiden Sektoren jeweils in eine vorbestimmte Anzahl gleich großer Blöcke (BLOCK0, ..., BLOCK3) geteilt werden und jeder Block in eine vorbestimmte Anzahl von Seiten, und der emulierte EEPROM-Speicherabschnitt aktualisiert wird, indem verschiedene Speicherstellen in einem Einzelbitmodus programmiert werden, so dass bei jeder Seitenaktualisierung ausgewählte Seitendaten zum nächsten freien Block verschoben werden und beim Einschreiben von Daten in jeden einzelnen der Blöcke des ersten Sektors alle Seiten in den zweiten Sektor umgespeichert werden.
2. Integrierte Schaltung nach Anspruch 1, **dadurch gekennzeichnet, dass** die EEPROM-Byte-Veränderlichkeit über eine Hardware-Einrichtung emuliert wird.
3. Integrierte Schaltung nach Anspruch 1, **dadurch gekennzeichnet, dass** 8 KByte des Flash-Speicherabschnitts dazu verwendet werden, um 1 KByte eines EEPROM-Speicherabschnitts zu emulieren.
4. Integrierte Schaltung nach Anspruch 1, **dadurch gekennzeichnet, dass** eine Zustandsmaschine (15) zum Steuern eines Adressenzählers (20) vorgesehen ist, dessen Ausgang an einen internen Adressbus (21) angeschlossen ist, der innerhalb der Speichermakrozelle (1) verläuft, wobei der Adressenzähler (20) Steuersignale von der Zustandsmaschine (15) empfängt, um das Laden von festcodierten Adressen in flüchtige oder nicht flüchtige Register (25) zu steuern, die während einer Rücksetzphase oder durch die Zustandsmaschine (15) nach einer Aktualisierung des EEPROM vom Mikrocontroller gelesen und aktualisiert werden.

5. Intégré Schaltung nach Anspruch 4, **dadurch gekennzeichnet, dass** der Adressbus (21) an den Eingang eines RAM-Puffers (22) angeschlossen ist, der für die Seitenaktualisierung des EEPROM verwendet wird, das zwei zusätzliche Bytes (23, 24) umfasst, um während einer Phase der Seitenaktualisierung die Seitenadresse zu speichern.

6. Intégré Schaltung nach Anspruch 1, **dadurch gekennzeichnet, dass** Vorgänge im Flash-Speicher und EEPROM-Speicher über eine Registerschnittstelle (7) gesteuert werden, die konform mit dem Speicher (1) ist.

7. Verfahren zum Emulieren der Leistungsmerkmale eines EEPROM-Speicherbausteins, der in einer Speicherzelle (1) enthalten ist, die in eine integrierte Schaltung eingebettet ist, welche auch einen Mikrocontroller und einen Flash-EEPROM-Speicher umfasst, der durch eine vorbestimmte Anzahl von Sektoren (F0, F1, F2, F3, F4, F5) gebildet ist, wobei mindestens zwei Sektoren (E0, E1) der Flash-Speicherstruktur dazu verwendet werden, die EEPROM-Byte-Veränderlichkeit zu emulieren, indem die beiden Sektoren jeweils in eine vorbestimmte Anzahl gleich großer Blöcke (BLOCK0, ..., BLOCK3) geteilt werden und jeder Block in eine vorbestimmte Anzahl von Seiten, und der emulierte EEPROM-Speicherabschnitt aktualisiert wird, indem verschiedene Speicherstellen in einem Einzelbitmodus programmiert werden, wobei bei jeder Seitenaktualisierung ausgewählte Seitendaten zum nächsten freien Block verschoben werden und, wenn Daten in jeden einzelnen der Blöcke des ersten Sektors eingeschrieben wurden, alle Seiten in den zweiten Sektor umgespeichert werden.

Revendications

1. Circuit intégré comprenant un microcontrôleur, une macro-cellule de mémoire (1) incluant une mémoire Flash EEPROM composée d'un nombre prédéterminé de secteurs (F0, F1, F2, F3, F4, F5) et d'un moyen d'émulation des caractéristiques d'un dispositif de mémoire EEPROM incorporé dans ladite macro-cellule de mémoire (1) ;
dans lequel ledit moyen comprend au moins deux secteurs (E0, E1) de ladite mémoire Flash EEPROM, qui sont conçus pour servir à émuler la capacité d'altération des octets EEPROM en divisant chacun desdits deux secteurs en un nombre prédéterminé de blocs (BLOC 0, ..., BLOC 3) de même taille, chaque bloc étant divisé en un nombre prédéterminé de pages, et en mettant à jour la partie de mémoire EEPROM émulée en programmant différents emplacements de mémoire dans un mode bit unique, de telle sorte que, à chaque mise à jour des pages, les données de la page sélectionnée sont déplacées vers le bloc libre suivant et, lorsque les données sont écrites dans chacun des blocs dudit premier secteur, toutes les pages sont permutées dans ledit deuxième secteur.
2. Circuit intégré selon la revendication 1, **caractérisé en ce que** ladite capacité d'altération des octets EEPROM est émulée par un moyen matériel.
3. Circuit intégré selon la revendication 1, **caractérisé en ce que** 8 Ko de la partie de mémoire Flash sont utilisés pour émuler 1 Ko d'une partie d'une mémoire EEPROM.
4. Circuit intégré selon la revendication 1, **caractérisé en ce qu'**une machine à états (15) est fournie pour commander un compteur d'adresses (20) qui est connecté en sortie à un bus d'adresses interne (21) fonctionnant à l'intérieur de ladite macro-cellule de mémoire (1), ledit compteur d'adresses (20) recevant des signaux de contrôle envoyés par la machine à états (15) de manière à commander le chargement des adresses figées dans le code dans des registres volatiles ou non volatiles (25) lus et mis à jour par le microcontrôleur au cours d'une phase de réinitialisation ou par la machine à états (15) après une mise à jour EEPROM.
5. Circuit intégré selon la revendication 4, **caractérisé en ce que** ledit bus d'adresses (21) est connecté à l'entrée d'une zone tampon de la RAM (22), utilisée pour la mise à jour des pages de la EEPROM incluant deux octets supplémentaires (23, 24) pour stocker l'adresse des pages au cours d'une phase de mise à jour des pages.
6. Circuit intégré selon la revendication 1, **caractérisé en ce que** le fonctionnement des mémoires Flash et EEPROM sont commandées par l'intermédiaire d'une interface de registre (7) mappée dans la mémoire (1).
7. Procédé d'émulation des caractéristiques d'un dispositif de mémoire EEPROM incorporé dans une macro-cellule de mémoire (1) qui est noyée dans un circuit intégré comprenant également un microcontrôleur et incluant une mémoire Flash EEPROM composée d'un nombre prédéterminé de secteurs (F0, F1, F2, F3, F4, F5), au moins deux secteurs (E0, E1) de la structure de mémoire Flash étant utilisés pour émuler la capacité d'altération des octets EEPROM en divisant chacun desdits deux secteurs en un nombre prédéterminé de blocs (BLOC 0, ...,

EP 0 991 081 B1

BLOC 3) de même taille, chaque bloc étant divisé en un nombre prédéterminé de pages, et en mettant à jour la partie de mémoire EEPROM émulée en programmant différents emplacements de mémoire dans un mode bit unique, dans lequel, à chaque mise à jour des pages, les données de la page sélectionnée sont déplacées vers le bloc libre suivant et, lorsque les données ont été écrites dans chacun des blocs dudit premier secteur, toutes les pages sont permutées dans ledit deuxième secteur.

5

10

15

20

25

30

35

40

45

50

55

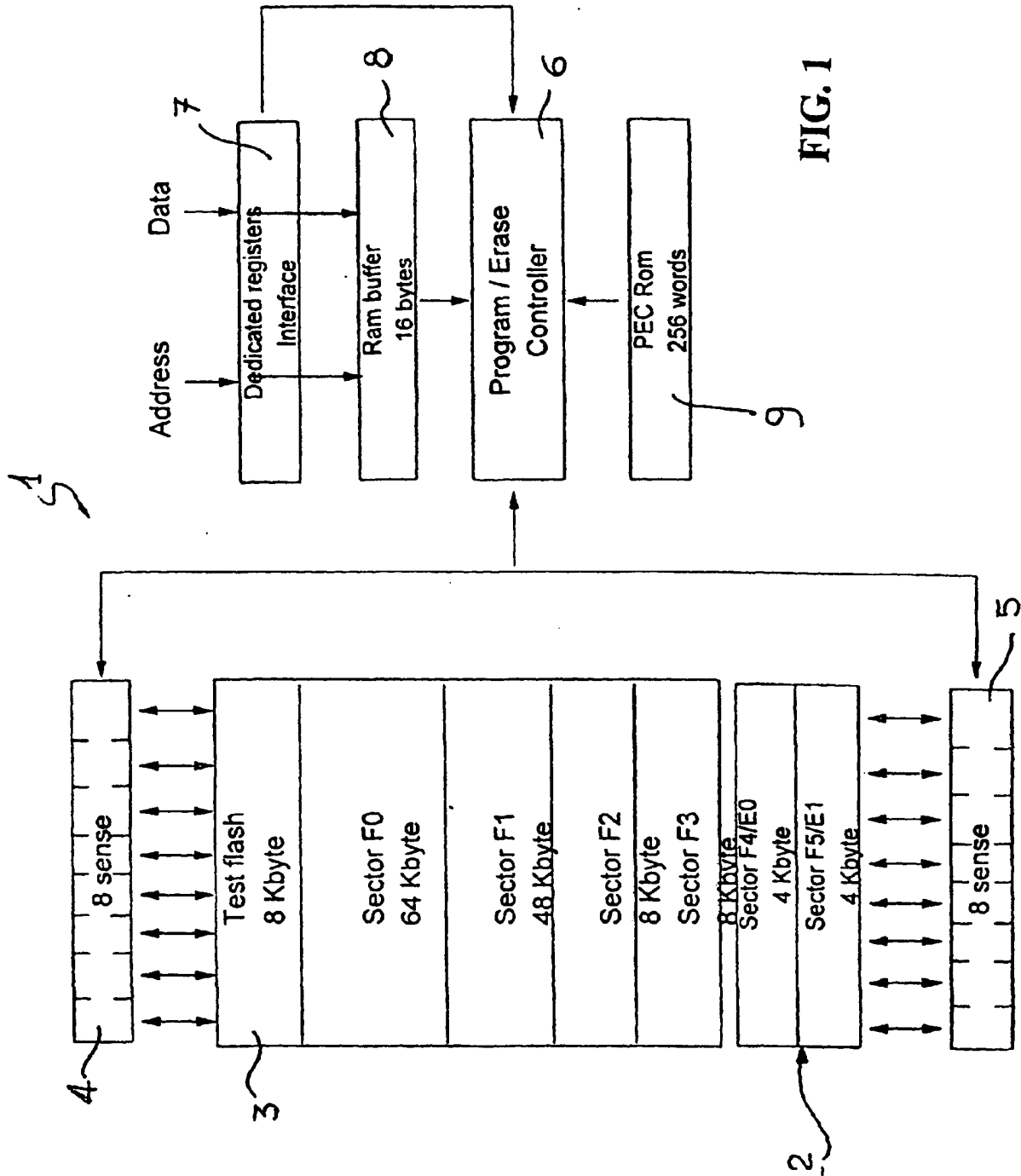
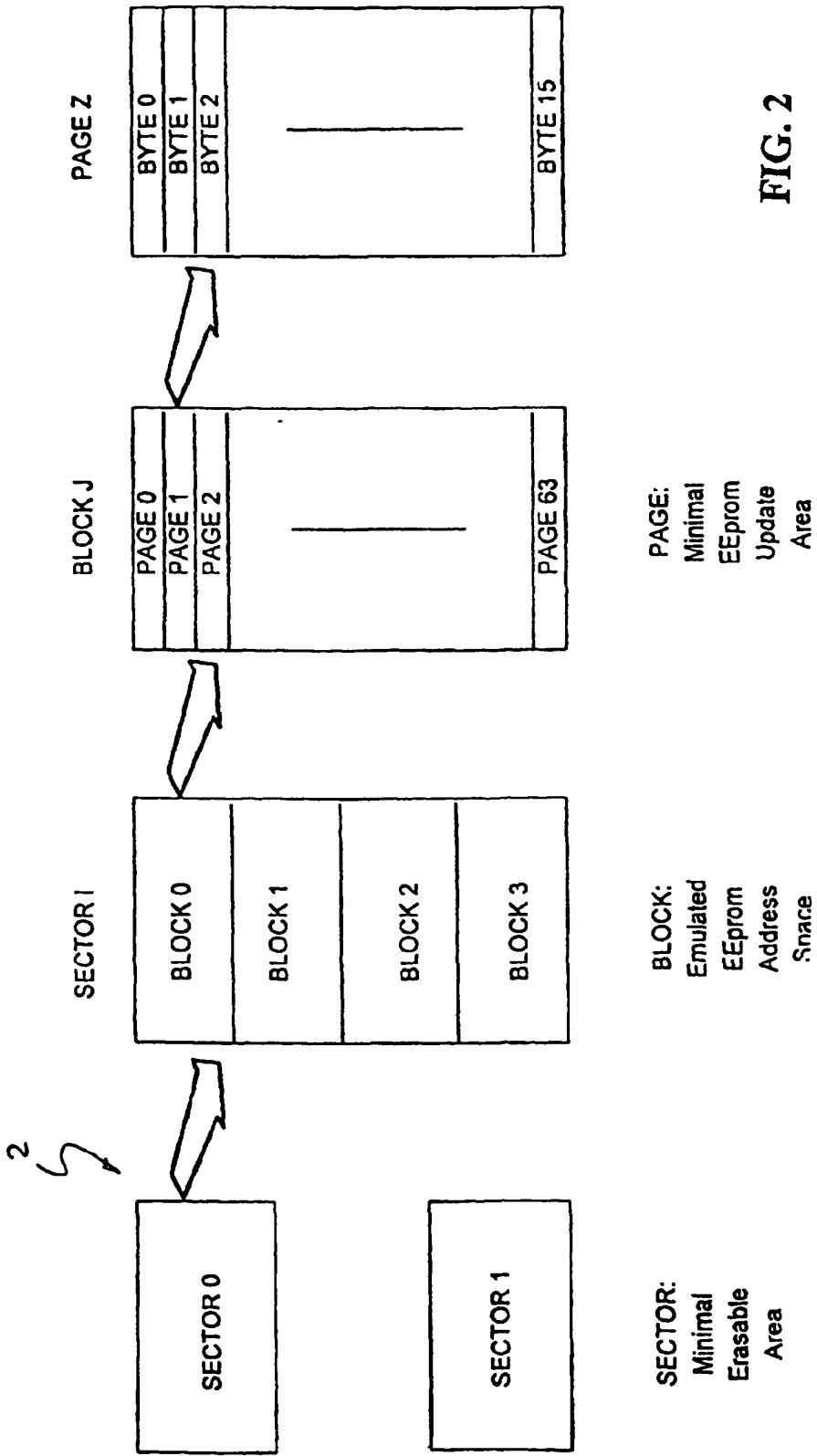


FIG. 1



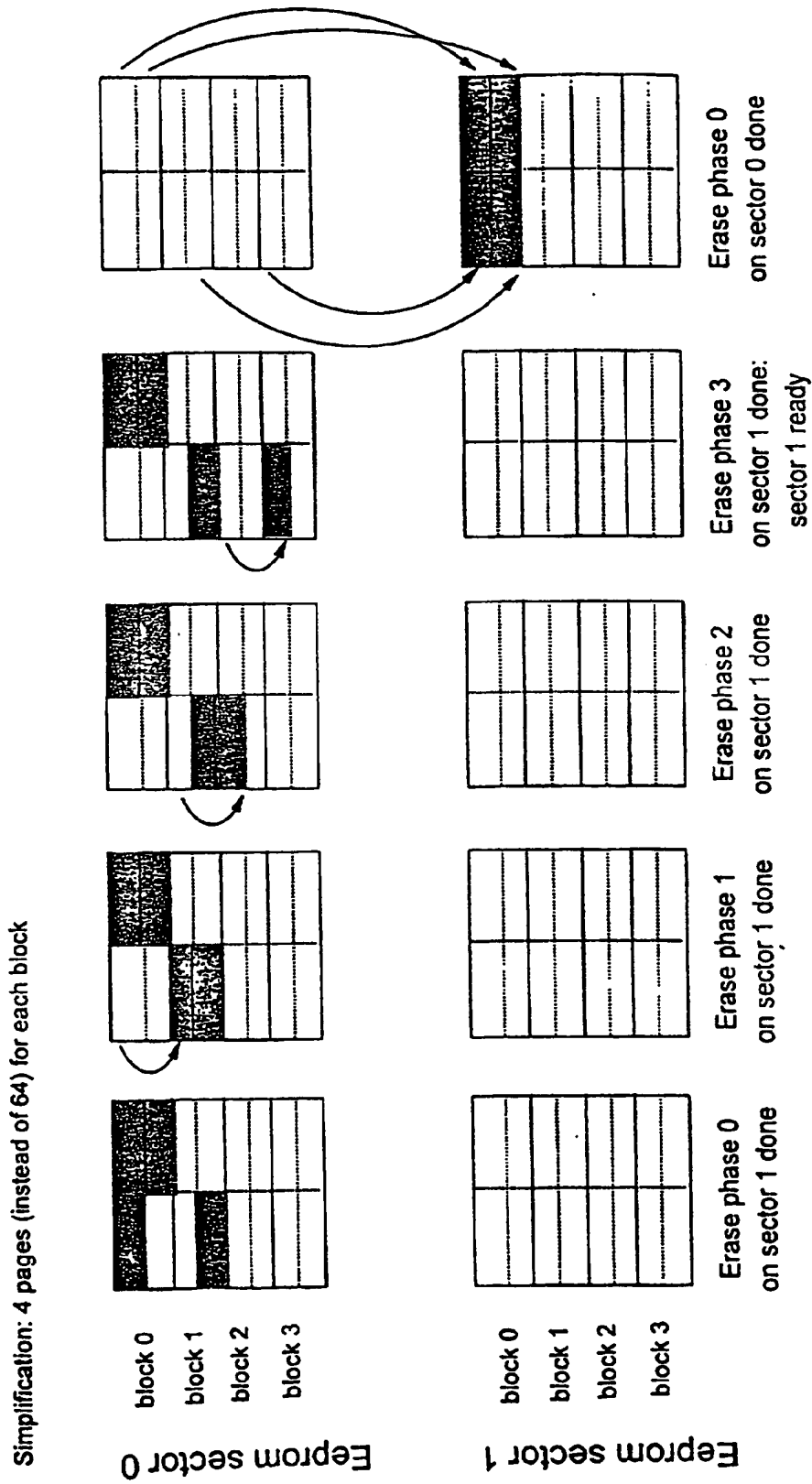


FIG. 3

| | |
|---------|-----|
| 224000h | FCR |
| 224001h | ECR |
| 224002h | FSR |

FIG. 3A

| Sector | Addresses | Max Size |
|---------|--------------------|----------|
| OTP | 211F80h to 211FFFh | 128 byte |
| Flash 0 | 000000h to 00FFFFh | 64 Kbyte |
| Flash 1 | 010000h to 01BFFFh | 48 Kbyte |
| Flash 2 | 01C000h to 01DFFFh | 8 Kbyte |
| Flash 3 | 01E000h to 01FFFFh | 8 Kbyte |
| Eeprom | 220000h to 2203FFh | 1 Kbyte |

FIG. 3B



FIG. 3C

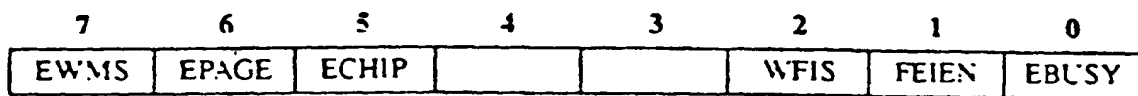


FIG. 3D

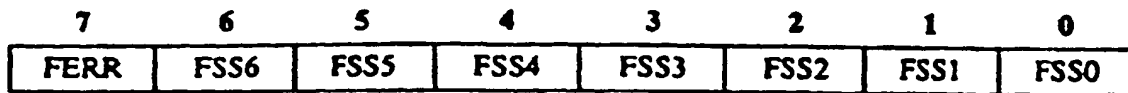


FIG. 3E

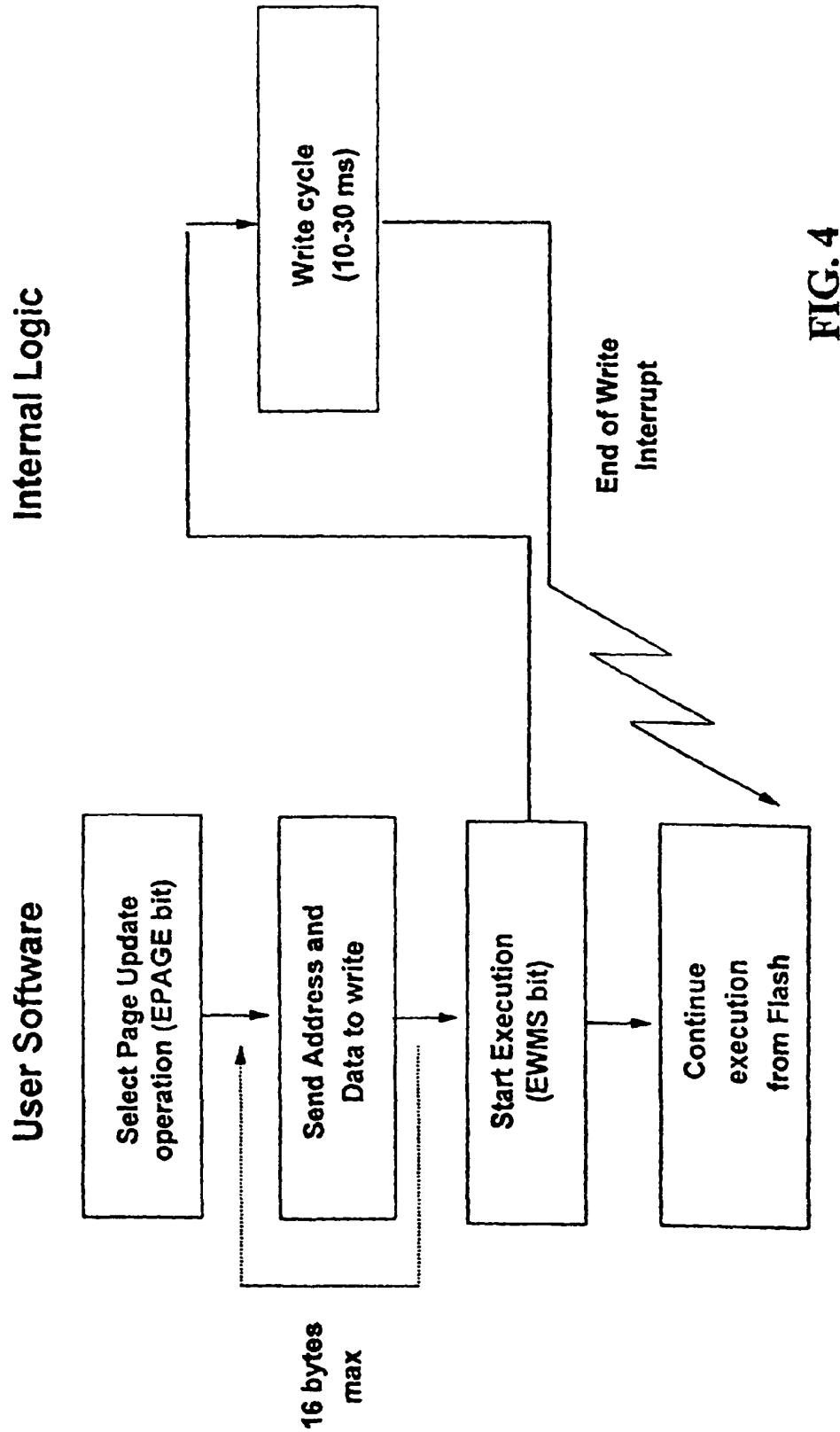


FIG. 4

| | |
|---------|--------|
| 211FFCh | NVAPR |
| 211FFDh | NVWPR |
| 211FFEh | NVPWD0 |
| 211FFFh | NVPWD1 |

FIG. 4A

| Operation | Size | Min | Typ | Max |
|-------------|----------|--------|-------|--------|
| Page Update | 256 byte | 160 us | 10 ms | 30 ms |
| | 512 byte | 160 us | 15 ms | 50 ms |
| | 1 Kbyte | 160 us | 30 ms | 100 ms |
| Chip Erase | 256 byte | | 35 ms | 100 ms |
| | 512 byte | | 45 ms | 150 ms |
| | 1 Kbyte | | 70 ms | 300 ms |

FIG. 4B

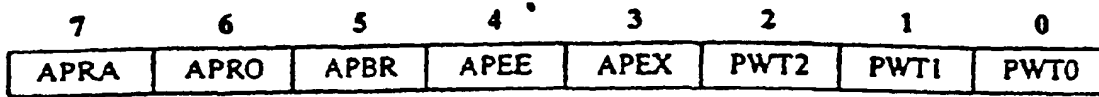


FIG. 4C

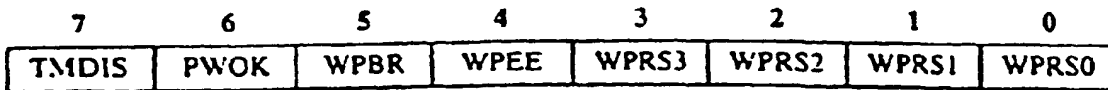


FIG. 4D

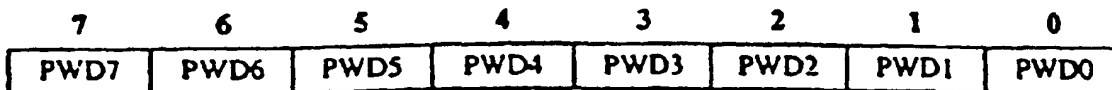


FIG. 4E

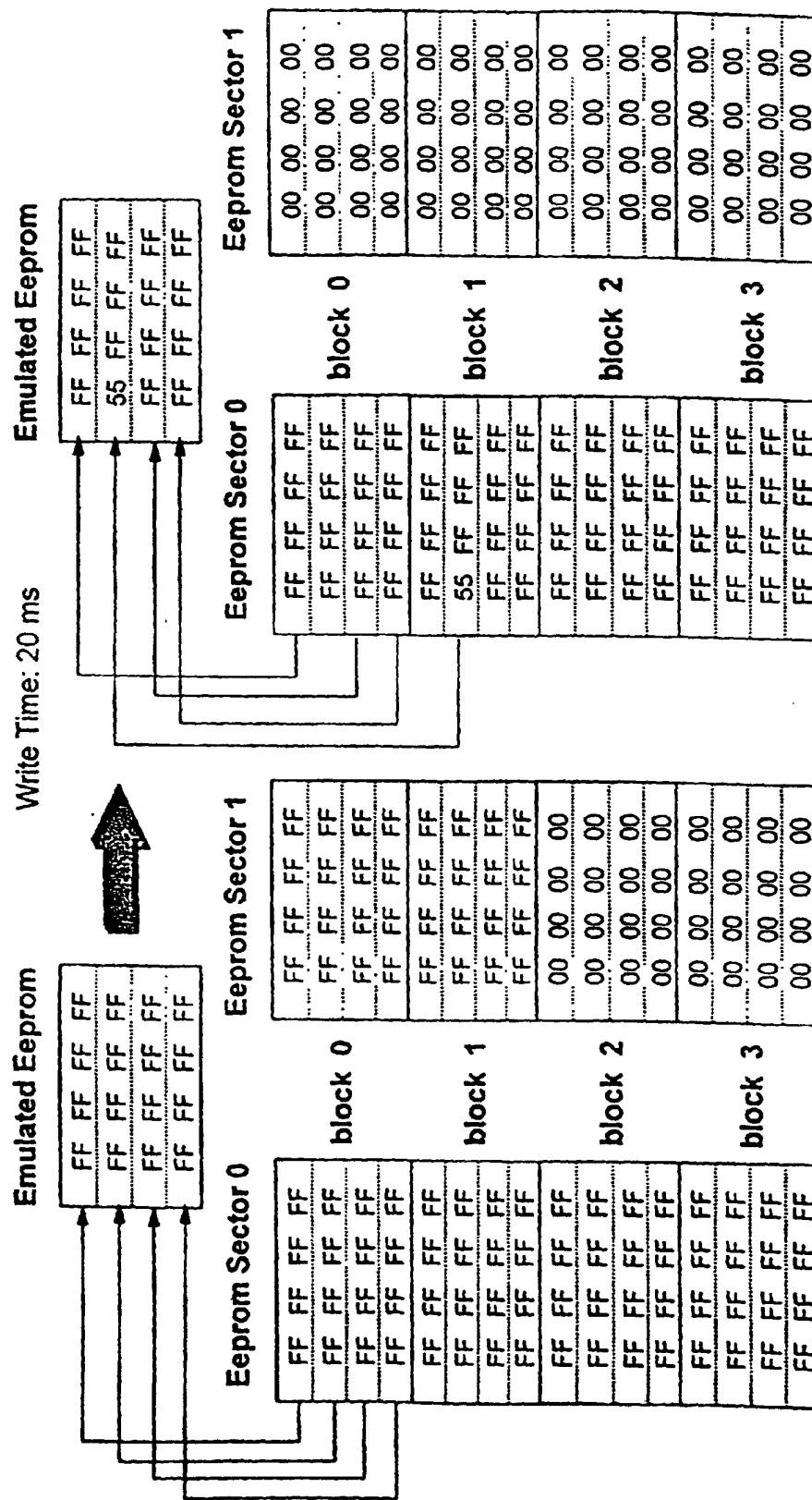


FIG. 5

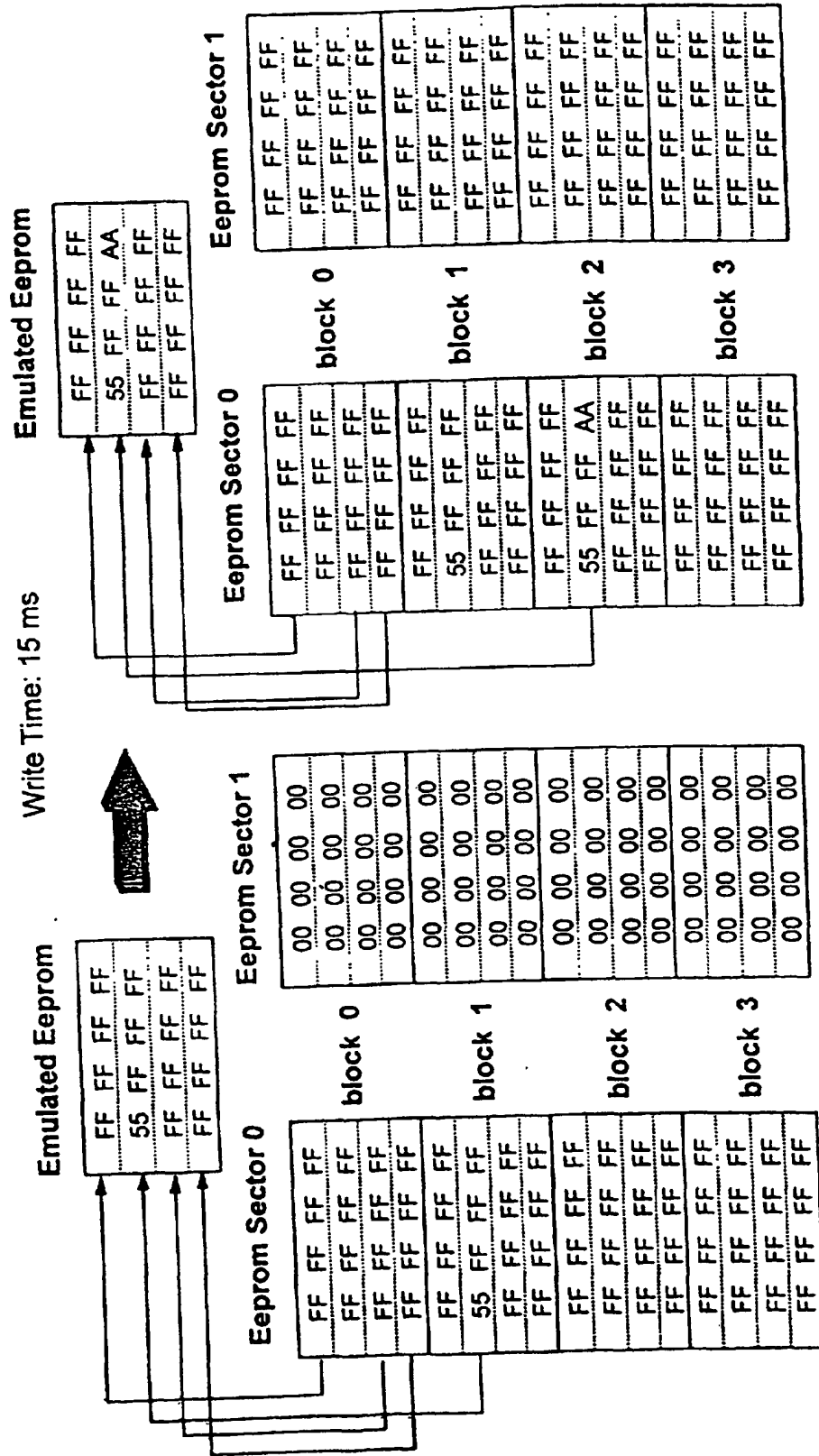


FIG. 6

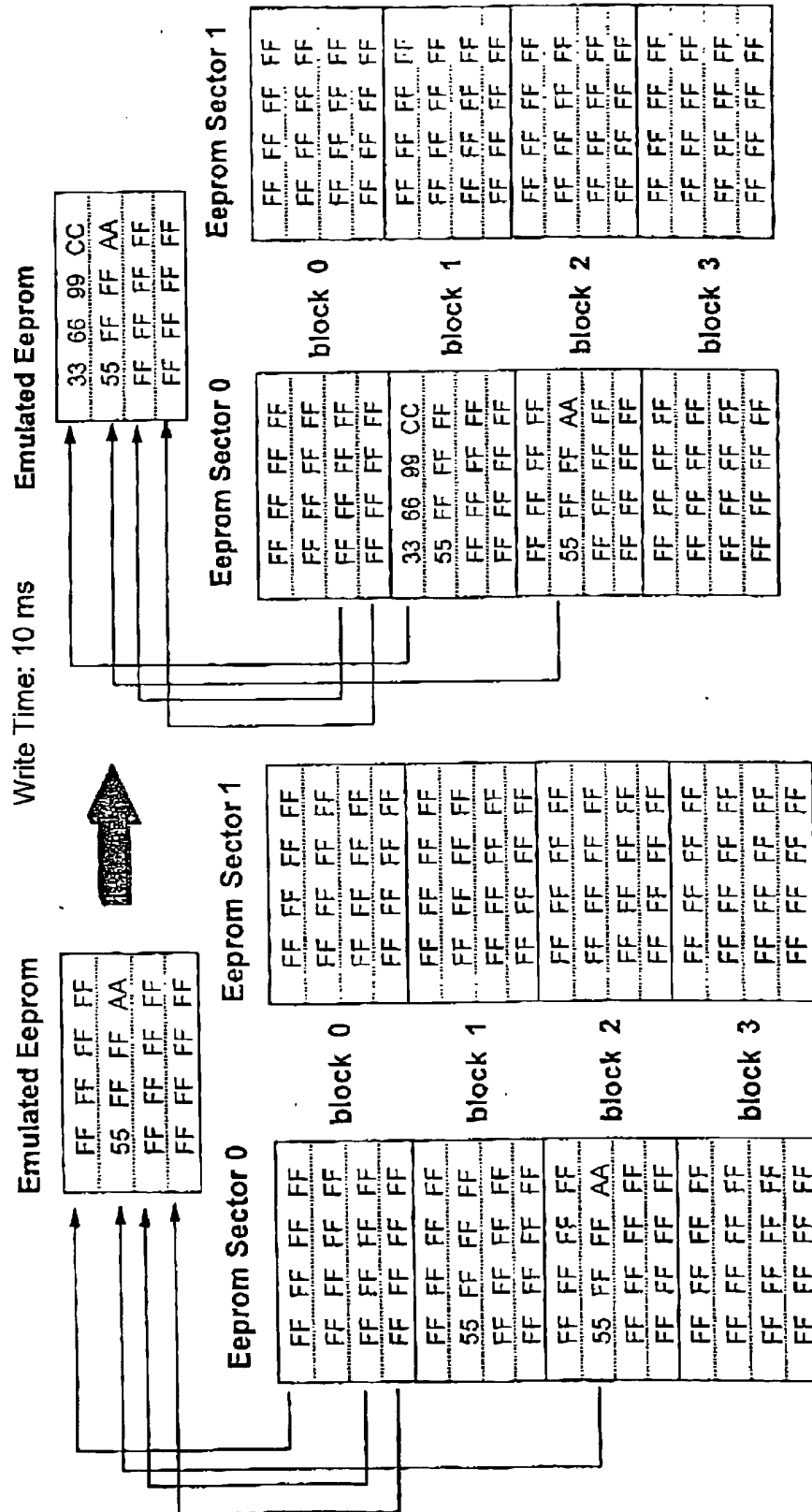


FIG. 7

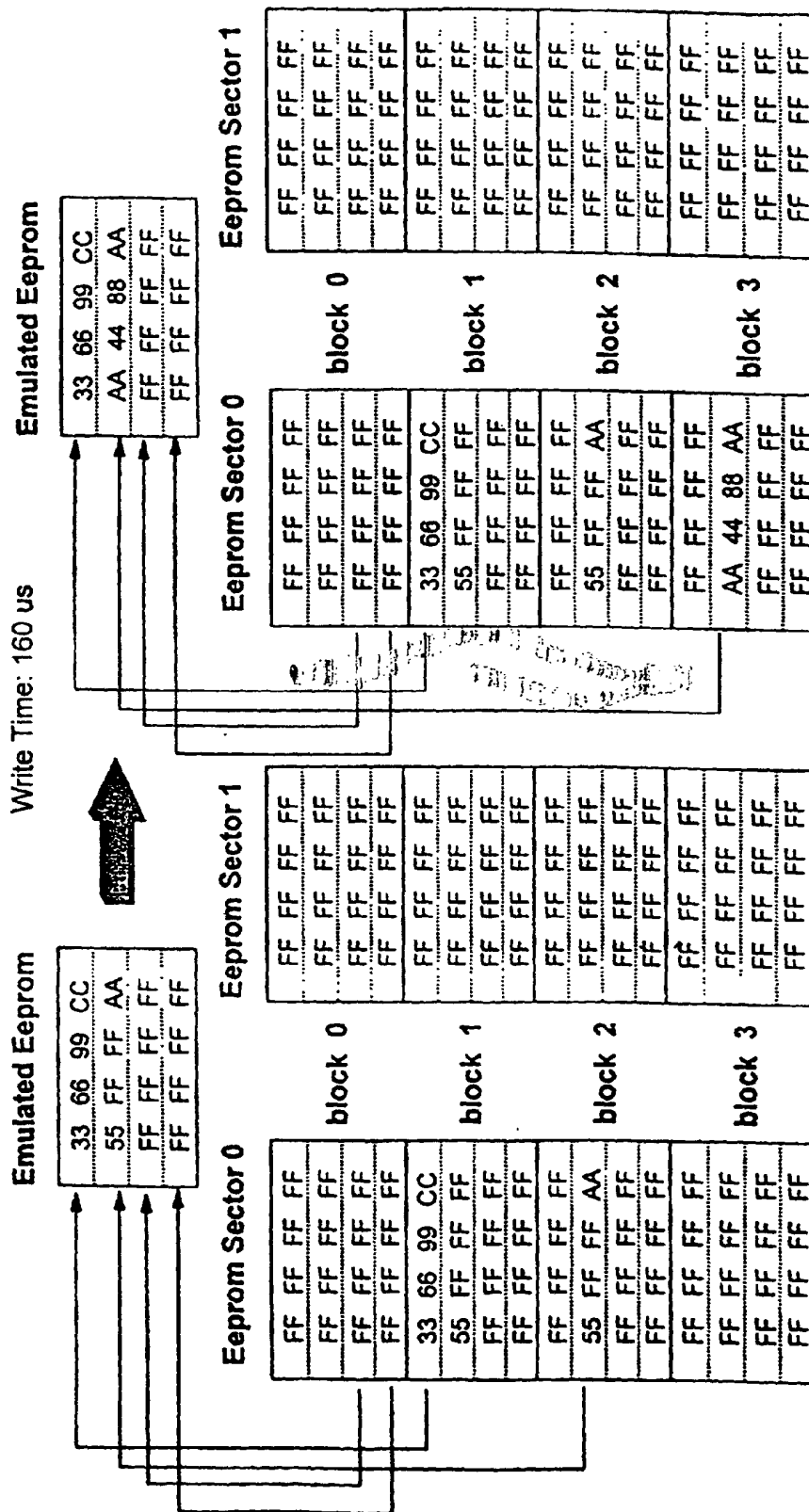


FIG. 8

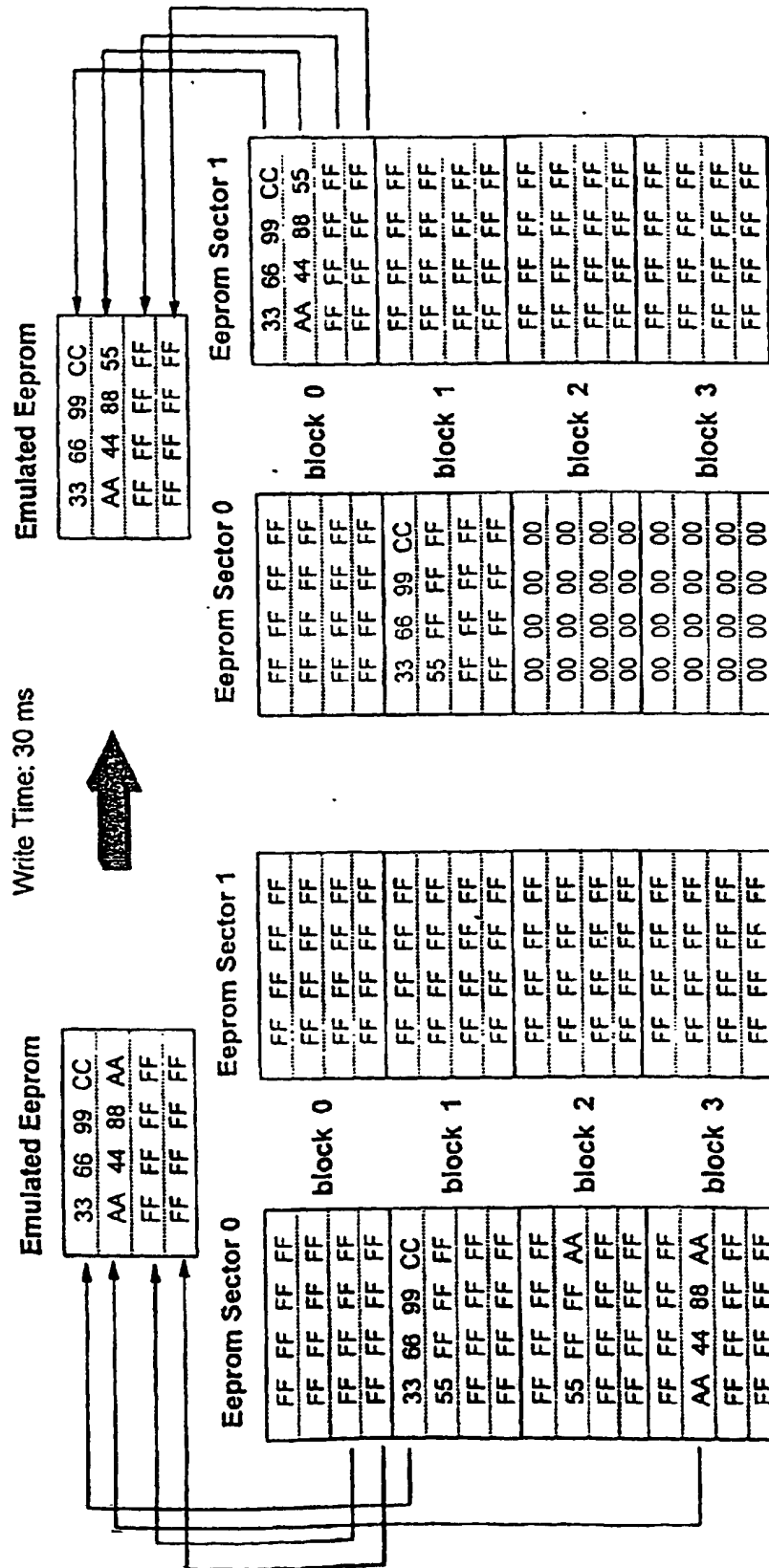


FIG. 9

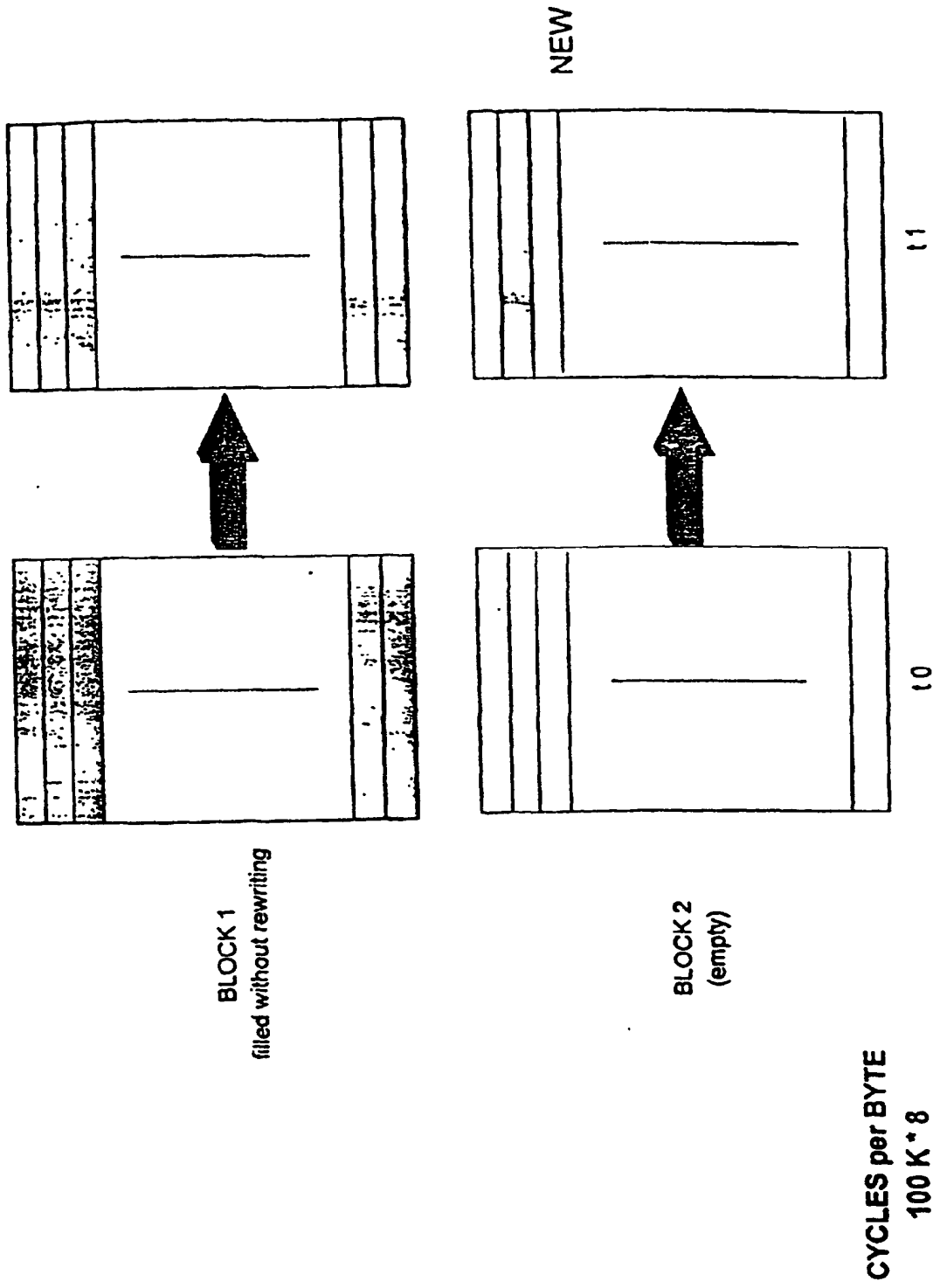


FIG. 10

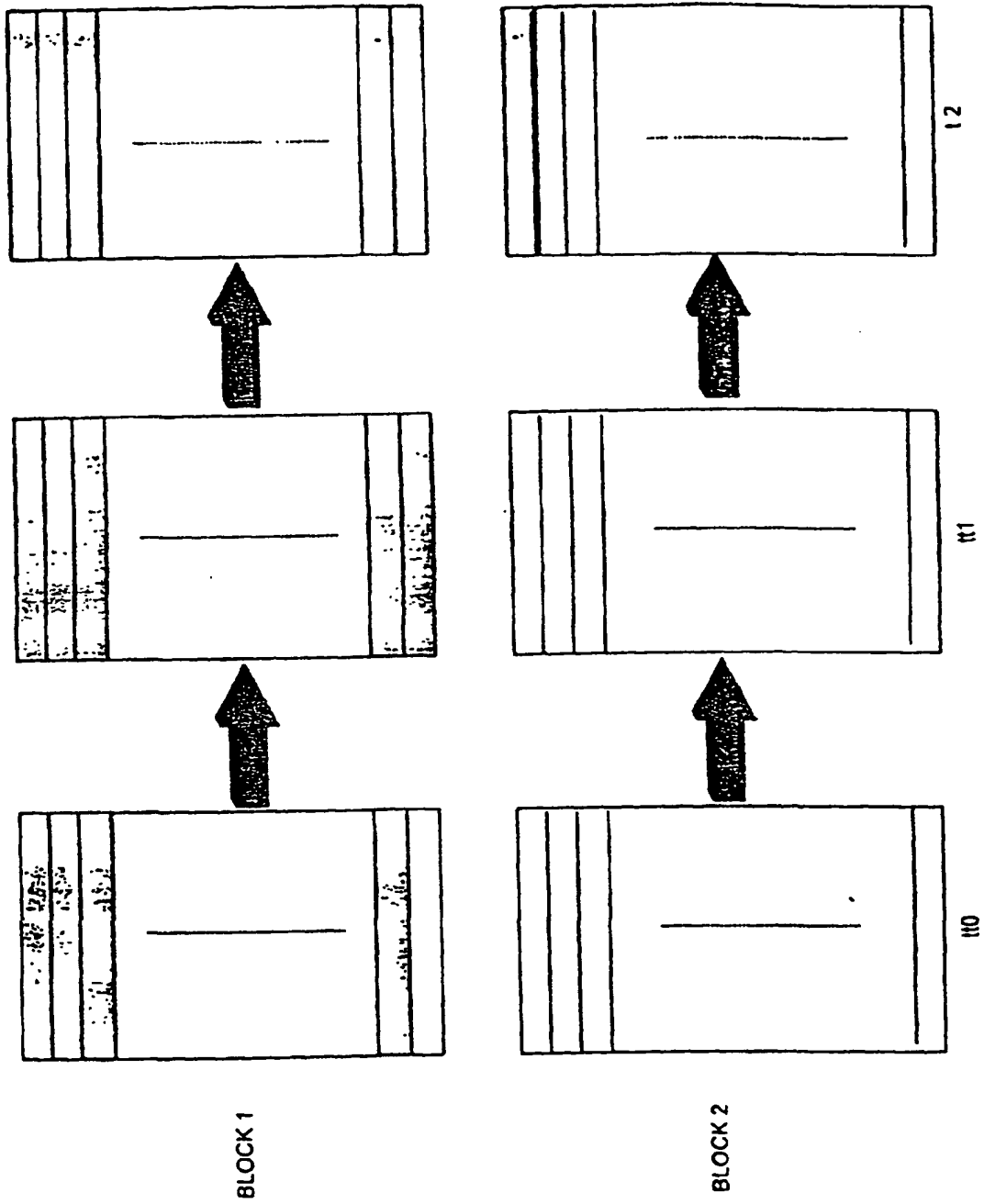


FIG. 11

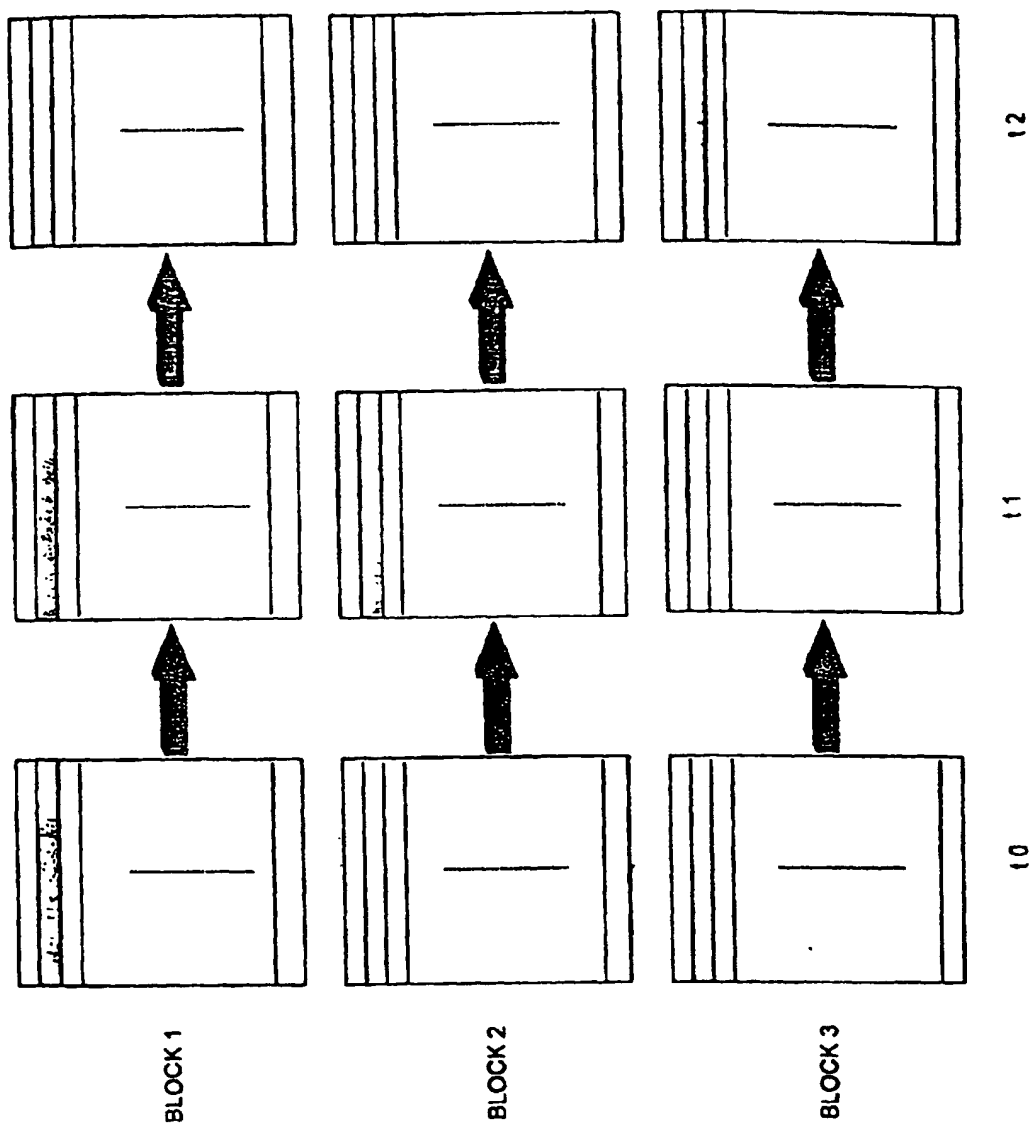


FIG. 12

FIG. 13

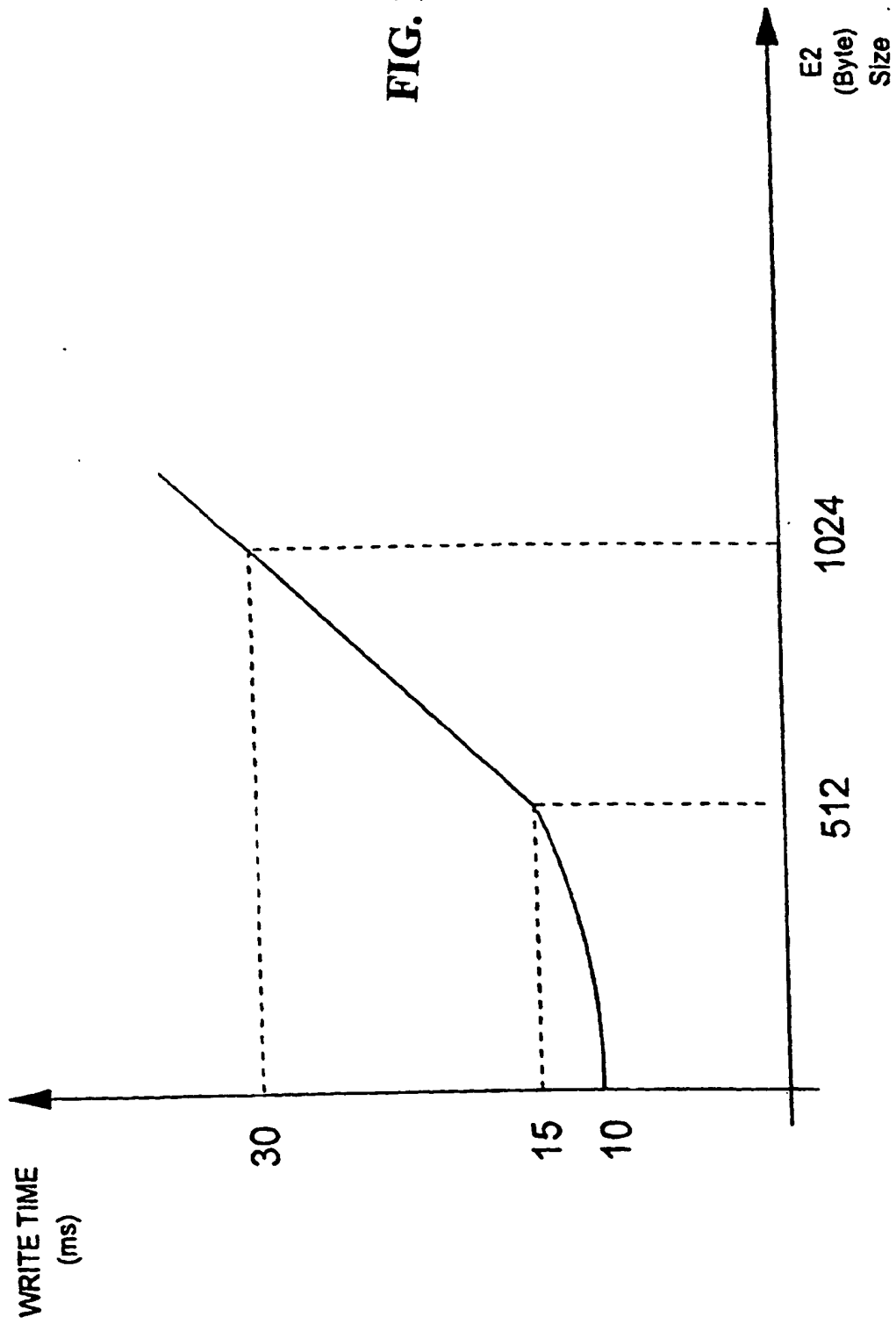


Fig. 14

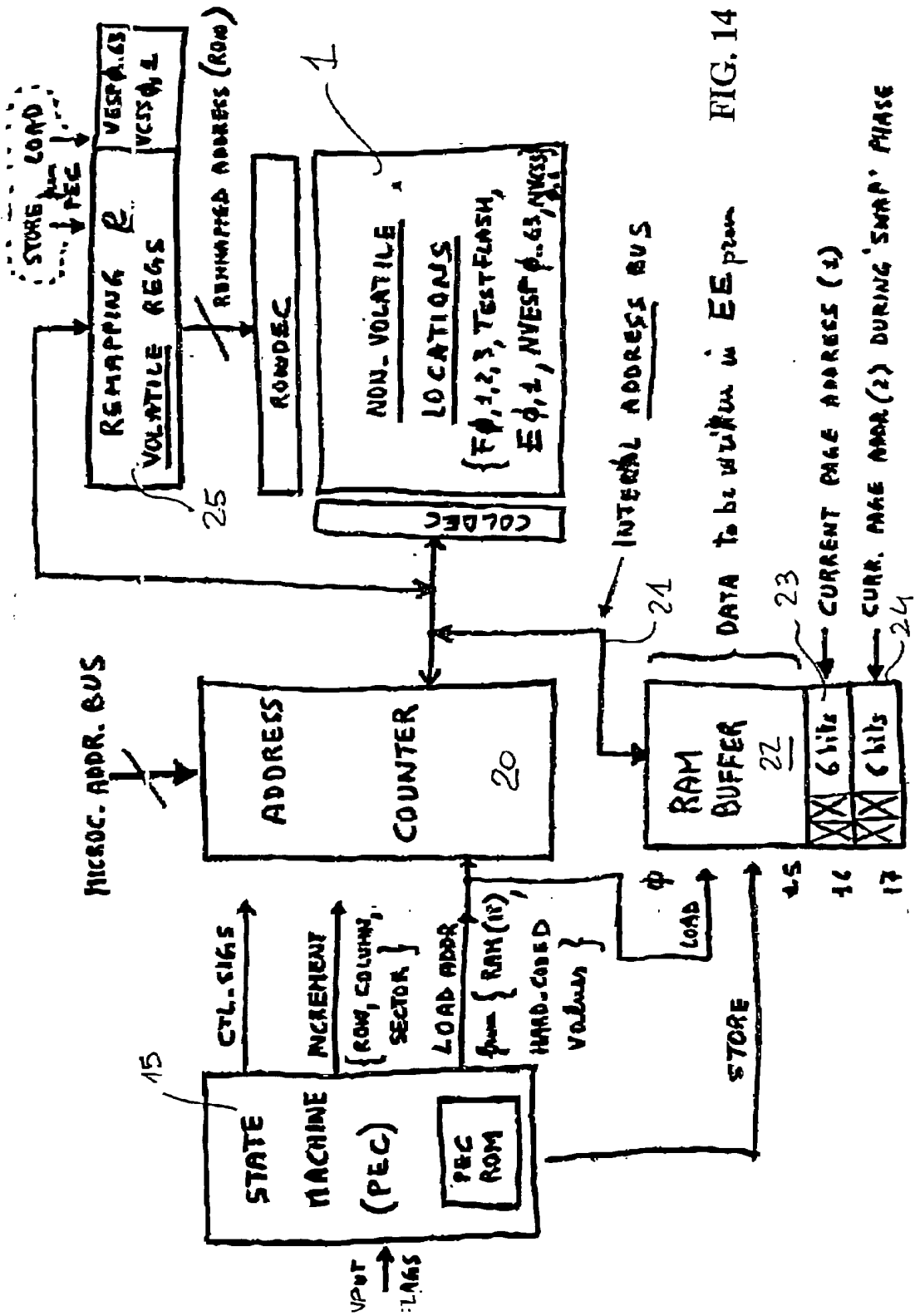


FIG. 14